深圳市紫光同创电子有限公司
SHENZHEN PANGO MICROSYSTEMS CO.,LTD

Titan2 Series FPGA Configuration User Guide

(UG050005, V1.0)

(2021.6.30)

Document version revision record

VERSION NUMBER	RELEASE DATE	REVISION RECORD
V0.1	2020/10/9	Initial Version
V0.2	2021/1/4	1. Add eFuse function bit lock description 2. Modify Figures 9, 10, 14 and 17 to update the reset wait time 3. Modify Figure 24 to reflect RSTN and INIT_FLAG_N, DONE delay 4. Modify the device name 330 to 390
V0.3	2021/4/6	1. Modify the multi-start fallback condition, add authentication failure 2. Add the description from not continuous loading 3. Add the description of the cascade timing requirements from the string 4. Add configuration time description 5. Modify Figure 25 Wake-up Timing 6. Add the description of the recommended settings for wake-up time sequence 7. Add all IO status tables 8. Add MODE built-in weak pull-up description 9. Modify the watchdog description
V1.0	2021/6/30	1. Add description "Configuration level selection" 2. Delete DCI description 3. Modify SEU support interface description 4. Modify Slave Parallel, Slave Serial frequency 5、 Add cascading bitstream tool generation location 6. Modify the gold bit flow initialization system description 7. Modify the application bitstream initialization system description 8. Delete readback CRC chapters 9. Modify SEU detection section 10. Modify the internal slave and interface application description 11. Modify the internal slave and interface parameters 12. Modify the description of the wake-up timing proposal and add the description of the SCBV grounding scenario

Terminology explanation

ABBREVIATIONS	FULL SPELLING
JTAG	Joint Test Action Group
SED	Soft Error Detection
CCS	Configuration Control System
SPI	Serial Peripheral Interface
AES	Advanced Encryption Standard
SEU	Single Event Upsets
ECC	Error Control Coding
SECDED	Single Error Correcting and Double Error
UID	Unique ID

Table of Contents

Figure Catalog

List of tables

I. Overview

Configuration is the process of writing the user's design data (bitstream) into the internal memory of the FPGA. Titan2 series FPGAs use SRAM cells to store configuration data, which is lost after power down, so the FPGA needs to be reconfigured every time it is powered up.

The Titan2 family of FPGAs supports the following four configuration modes:

- JTAG mode, compliant with IEEE 1149.1, IEEE 1149.6 and IEEE 1532 standards
- Master SPI mode, supports data bit width 1/2/4/8bit
- Slave Parallel mode, supports data bit width 8/16/32bit
- Slave Serial mode, supports data bit width 1bit

In addition, the Titan2 family of FPGAs offers the following features:

- Watchdog with timeout detection support.
- Supports SEU 1-bit error correction and 2-bit error detection via internal slave parallel interface
- Supports configuration of bitstream version fallback in Master SPI mode
- Configure bitstream compression to effectively reduce bitstream size, storage space and programming time
- Configure bitstream encryption to prevent malicious copying and effectively protect customer design intellectual property
- JTAG mode provides a dedicated interface to support in-circuit debugging and boundary-scan testing
- Each device is factory written with a unique code that corresponds to it - 96bits of UID protection
- Support SHA digest, RSA-2048 authentication, AES256-GCM self-authentication for digital signature verification of the bitstream to verify the integrity of the bitstream
- Key storage method supports eFuse and BB-RAM, with BB-RAM providing chip-level security
- Security technology that supports anti-bitstream reverse read
- Support JTAG security management, can permanently or temporarily close the JTAG function
- Supports anti-DPA attacks to prevent encryption keys from being cracked by hackers

II. Configuration mode description

Users can select the configuration mode of Titan2 series FPGAs by setting the value of MODE[2:0], as shown in Table 1. Among them, JTAG configuration mode has the highest priority, and MODE[2:0] set to any value (except 3'b000) can be configured. The direction of the configuration clock CFG_CLK is determined by the configuration mode. In master mode, CFG_CLK is the output, which is output by the FPGA to an external device that stores the bit stream, such as Flash; in slave mode, CFG_CLK is the input, which is output by an external device (such as a microprocessor, CPLD or other FPGA) to the FPGA.

Table 1 Titan2 Series FPGA Configuration Mode

SERIAL NUMBER	CONFIGURATION MODE	MODE[2:0] (1)	DATA BIT WIDTH	CFG_CLK DIRECTION
1	JTAG	xxx(2)	1	Input (TCK)
2	Master SPI	001	1,2,4,8	Output
3	Slave Parallel	110	8,16,32	Input
4	Slave Serial	111	1	Input

Note: (1) MODE has built-in weak pull-up, i.e. if there is no external pull-up resistor, MODE[2:0]=3'b111, i.e. slave string mode.

(2) prohibit the setting of MODE[2:0]=3'b000, this mode any interface can not be used. In addition to 3'b000, set to any value can use JTAG mode, if only need to use JTAG mode, can be set to MODE[2:0]=3'b101, only JTAG available under this setting.

The pins used in different configuration modes are shown in the table below, and some pins will have different functions in different configuration modes.

Table 2 Serial Mode Configuration Pins

NAME	BANK	PROPERTY	JTAG	MASTER SPI				SLAVE SERIAL
				X1	X2	X4	X8	
SCBV	VCCIOCFG	Dedicated	SCBV	SCBV	SCBV	SCBV	SCBV	SCBV
MODE[2:0]	VCCIOCFG	Dedicated	xxx(1)	001	001	001	001	111
TCK	VCCIOCFG	Dedicated	TCK	TCK	TCK	TCK	TCK	TCK
TMS	VCCIOCFG	Dedicated	TMS	TMS	TMS	TMS	TMS	TMS
TDI	VCCIOCFG	Dedicated	TDI	TDI	TDI	TDI	TDI	TDI
TDO	VCCIOCFG	Dedicated	TDO	TDO	TDO	TDO	TDO	TDO
RSTN	VCCIOCFG	Dedicated	RSTN	RSTN	RSTN	RSTN	RSTN	RSTN
INIT_FLAG_N	VCCIOCFG	Dedicated	INIT_FLAG_N	INIT_FLAG_N	INIT_FLAG_N	INIT_FLAG_N	INIT_FLAG_N	INIT_FLAG_N
CFG_DONE	VCCIOCFG	Dedicated	CFG_DONE	CFG_DONE	CFG_DONE	CFG_DONE	CFG_DONE	CFG_DONE
CFG_CLK	VCCIOCFG	Dedic	-	CFG_CLK	CFG_CLK	CFG_CLK	CFG_CLK	CFG_CLK

		ated						
IO_STATUS_C	VCCIOL5	Multiplexing	IO_STATUS_C	IO_STATUS_C	IO_STATUS_C	IO_STATUS_C	IO_STATUS_C	IO_STATUS_C
ECCLKIN	VCCIOL5	Multiplexing	-	ECCLKIN	ECCLKIN	ECCLKIN	ECCLKIN	-
FCS_N	VCCIOL5	Multiplexing	-	FCS_N	FCS_N	FCS_N	FCS_N	-
FCS2_N	VCCIOL4	Multiplexing	-	-	-	-	FCS2_N	-
CS_N	VCCIOL5	Multiplexing	-	-	-	-	-	-
RWSEL	VCCIOL5	Multiplexing	-	-	-	-	-	-
D[31:16]	VCCIOL5	Multiplexing	-	-	-	-	-	-
D[15:8]	VCCIOL5	Multiplexing	-	-	-	-	-	-
D[7:4]	VCCIOL5	Multiplexing	-	-	-	-	D[7:4]	-
D[3:2]	VCCIOL5	Multiplexing	-	-	-	D[3:2]	D[3:2]	-
MISO_D1_DI	VCCIOL5	Multiplexing	-	MISO	D[1]	D[1]	D[1]	DI
MOSI_D0	VCCIOL5	Multiplexing	-	MOSI	D[0]	D[0]	D[0]	-
CSO_DOUT	VCCIOL5	Multiplexing	-	CSO_DOUT	-	-	-	CSO_DOUT

Note: (1) It is forbidden to set MODE[2:0]=3'b000, any interface in this mode cannot be used. In addition to 3'b000, set it to any value to use JTAG mode. If you only need to use JTAG mode, you can set it to MODE[2:0]=3'b101. Only JTAG is available under this setting.

Table 3 Parallel Mode Configuration Pins

NAME	BANK	PROPERTIES	SLAVE PARALLEL		
			X8	X16	X32
SCBV	VCCIOCFG	Dedicated	SCBV	SCBV	SCBV
MODE[2:0]	VCCIOCFG	Dedicated	110	110	110
TCK	VCCIOCFG	Dedicated	TCK	TCK	TCK

TMS	VCCIOCFG	Dedicated	TMS	TMS	TMS
TDI	VCCIOCFG	Dedicated	TDI	TDI	TDI
TDO	VCCIOCFG	Dedicated	TDO	TDO	TDO
RSTN	VCCIOCFG	Dedicated	RSTN	RSTN	RSTN
INIT_FLAG_N	VCCIOCFG	Dedicated	INIT_FLAG_N	INIT_FLAG_N	INIT_FLAG_N
CFG_DONE	VCCIOCFG	Dedicated	CFG_DONE	CFG_DONE	CFG_DONE
CFG_CLK	VCCIOCFG	Dedicated	CFG_DONE	CFG_DONE	CFG_DONE
IO_STATUS_C	VCCIOL5	Multiplexing	IO_STATUS_C	IO_STATUS_C	IO_STATUS_C
ECCLKIN	VCCIOL5	Multiplexing	-	-	-
FCS_N	VCCIOL5	Multiplexing	-	-	-
FCS2_N	VCCIOL4	Multiplexing	-	-	-
CS_N	VCCIOL5	Multiplexing	CS_N	CS_N	CS_N
RWSEL	VCCIOL5	Multiplexing	RWSEL	RWSEL	RWSEL
D[31:16]	VCCIOL5	Multiplexing	-	-	D[31:16]
D[15:8]	VCCIOL5	Multiplexing	-	D[15:8]	D[15:8]
D[7:4]	VCCIOL5	Multiplexing	D[7:4]	D[7:4]	D[7:4]
D[3:2]	VCCIOL5	Multiplexing	D[3:2]	D[3:2]	D[3:2]
MISO_D1_DI	VCCIOL5	Multiplexing	D[1]	D[1]	D[1]
MOSI_D0	VCCIOL5	Multiplexing	D[0]	D[0]	D[0]
CSO_DOUT	VCCIOL5	Multiplexing	CSO_DOUT	CSO_DOUT	CSO_DOUT

The configuration pin functions are defined in the following table:

Table 4 Configuration Pin Function Definition

NAME	BANK	PROPERTIES	I/O	DESCRIPTION
SCBV	VCCIOCFG	Dedicated	input	Select the bank voltage where the multiplexing configuration pins are located. Use this pin to determine the voltage difference between BANKCFG and the voltage of the banks containing the alternate configuration pins, and the I/O voltage operating range for those banks. This pin is always active for BANKCFG, but only during configuration for Banks that contain alternate configuration pins. When the voltage of VCCIOCFG is 2.5V or 3.3V, this pin must be connected to high level (it can be directly connected to VCCIOCFG). When the voltage of VCCIOCFG is 1.8V or lower, this pin must be connected to low level (can be directly connected to ground). For the usage of different configuration modes and configuration voltages, see Configuration Level Selection.
MODE[2:0]	VCCIOCFG	Dedicated	input	Configuration-specific input pin for configuration mode selection; built-in weak pull-up resistor; In the determination mode, you can use a resistor less than 1KOhm to pull it up to VCCIOCFG or use a resistor less than 1KOhm to pull it down to VSS.
TCK	VCCIOCFG	Dedicated	input	IEEE STD 1149.1 compliant test clock input pin; Providing clocks for the FPGA's JTAG chain and testing the clock input pins;

				Built-in weak pull-up resistor (always enable), pull-up to VCCIOCFG
TMS	VCCIOCFG	Dedicated	input	Dedicated JTAG, test mode selection input pin; Built-in weak pull-up resistor (always enable), pull-up to VCCIOCFG
TDI	VCCIOCFG	Dedicated	input	Dedicated JTAG, test data input pin; Built-in weak pull-up resistor (always enable), pull-up to VCCIOCFG
TDO	VCCIOCFG	Dedicated	output	Dedicated JTAG, test data output pin; Built-in weak pull-up resistor (always enable), pull-up to VCCIOCFG
RSTN	VCCIOCFG	Dedicated	input	Dedicated configuration reset pin with built-in weak pull-up resistor and the weak pull-up is always active; used to reset the configuration logic and configuration memory, low active. When this pin is set low, the FPGA configuration memory is cleared and a new configuration process begins. The configuration logic reset begins on the falling edge of this pin, and the configuration process begins on the rising edge immediately after this pin. This pin requires an external pull-up resistor no larger than 4.7KΩ to VCCIOCFG. Holding this pin low at power-up does not put the FPGA configuration logic in a reset state. After configuration is complete, the user can optionally configure it as a weak pull-up or float.
INIT_FLAG_N	VCCIOCFG	Dedicated	Bidirectional (open-drain)	Pin dedicated to initialization and configuration status. When the output is low, it indicates that the FPGA chip is performing an initialization operation (clearing configuration memory) or a configuration error has occurred. Built-in weak pull-up resistors to enable weak pull-ups during configuration; When FPGA power-up is complete, drive this pin low. When the FPGA initialization is complete, the pin is released from drive. This pin can be driven low by an external input during power-up and during initialization to delay the configuration process after initialization is complete. When the chip detects a high input on this pin after the initialization is complete, the chip starts the configuration process. During the configuration process, this pin is used as an indicator status output for configuration errors, and a low indicates an error in the configuration process; This pin requires an external weak pull-up resistor to VCCIOCFG (recommended resistance no greater than 4.7KΩ) After the configuration is complete, the user can optionally configure it as a weak pull-up or float.

CFG_DONE	VCCIOCFG	Dedicated	Bidirectional (open-drain)	Configuration status-specific pins. Built-in weak pull-up resistor (approx. 10K Ω); Output as a configuration completion indication status, high indicates configuration completion. The default is an open-drain output. When FPGA power-up is complete, this pin is driven low prior to or during configuration. Once all configuration data has been properly received and wake-up timing has been initiated, control of this pin is released. After the configuration is complete, the pin can continue to be driven low externally. Once the internal wake-up timing finds that the external DONE pin is low, then the internal wake-up circuit continues to hold until the external pin goes high before continuing the wake-up process. After the configuration is complete, the user can optionally configure it as a weak pull-up or float.
CFG_CLK	VCCIOCFG	Dedicated	Input/output	The configuration clock pin. This clock is used to synchronize the FPGA configuration process in all modes except JTAG configuration mode. In Slave Serial, Slave Parallel mode, this pin is used as the clock input to obtain configuration data from external sources; In Master SPI configuration mode, this pin is used as a clock output to obtain configuration data externally, in this mode, an external 1kOhms pull-up resistor; When this clock is not needed (e.g. JTAG mode), it is in the high resistance state. After configuration is complete, the user can optionally configure it to be weak pull-up or floating.
IO_STATUS_C	VCCIOL5	Multiplexing	Input	Multiplexing pin, input signal, controls whether the weak pull-up resistor of all user IOs (including multiplexed IOs) is enabled from the completion of power-up until the user mode is entered. (1) "0" to enable all user IO internal pull-up resistors. (2) "1", does not enable all user IO internal pull-up resistors. (3) It is recommended to add weak pull-up resistors externally to this pin to the corresponding VCCIO. (4) This pin can be connected to the corresponding VCCIO or VSS either directly or through a resistor smaller than 1KΩ . (5) The pin is not allowed to dangle before or during configuration;
ECCLKIN	VCCIOL5	Multiplexing	Input	External input for the master mode configuration clock. This is an optional external clock input to the configuration logic. (1) In master mode (Master SPI), the FPGA can select this clock input as the configuration clock for the configuration logic. This clock can be divided (depending on the settings in the bitstream) and output from CFG_CLK. (2) For other configuration modes, usage modes, and during initialization, this pin will not be used as normal user I/O. for the high resistance state or weak pull-up.
FCS_N	VCCIOL5	Multiplexing	output	Multiplexed configuration pins, external Master SPI configuration mode use. (1) Master SPI X1, X2, X4 mode, low valid output chip select signal to external flash; also requires external pull-up resistor (no greater than 4.7KΩ) to VCCIO of the Bank where it is located (2) In other configuration modes and during initialization, this pin is used as a normal user I/O with a high resistance state or weak pull-up. (3) Once configured, it can be used as user I/O.

FCS2_N	VCCIOL4	Multiplexing	Output	Multiplexed configuration pins, external Master SPI X8 configuration mode use. (1) Master SPI X8 mode, low valid output chip select signal to external flash; requires external pull-up resistor no greater than 4.7KΩ to VCCIO (2) For other configuration modes, usage modes, and during initialization, this pin will not be used as a normal user I/O, a high resistance state or a weak pull-up. (3) Once configured, it can be used as user I/O.
CS_N	VCCIOL5	Multiplexing	input	Multiplexed configuration pin. Chip select input, active low. (1) Enables the Slave Parallel mode configuration interface for a low level. In Slave Parallel configuration mode, an external controller can control this pin to select the Slave Parallel bus of the FPGA that is about to operate, or to connect to the CSO_DOUT pin of the previous FPGA in the Slave Parallel configuration chain. (2) For other configuration modes and during initialization, this pin will not be used as a normal user I/O with a high resistance state or weak pull-up. (3) After the configuration is finished, it can be used as for I/O.
RWSEL	VCCIOL5	Multiplexing	input	Multiplexed configuration pins. slave Parallel configuration mode read/write select input, high for read, low for write. (1) High level, Slave Parallel configuration mode reads data from the data bus; (2) Low level, Slave Parallel configuration mode writes data to the data bus; (3) Switching between reads and writes can only be done while CS_N is high. (4) After configuration, it can be used as user I/O. (5) For other configuration modes and during initialization, this pin will not be used as a normal user I/O with a high resistance state or weak pull-up.
D[31:16]	VCCIOL5	Multiplexing	Input/Output	Multiplexing configuration data pins (1) For Slave Parallel, X32 bits wide, D[31:16] is used as the [31:16] bit of the data bus. (2) For other configuration modes, insufficient bit width and during initialization, this pin will not be used as normal user I/O, for high resistance or weak pull-up. (3) Once configured, it can be used as user I/O.
D[15:8]	VCCIOL5	Multiplexing	Input/Output	Multiplexed configuration data pins (1) For Slave Parallel mode, X16 bits wide, D[15:8] is used as bit [15:8] of the data bus, and for Slave ParallelX32 bits wide, D[15:8] is used as bit [15:8] of the data bus. (2) For other configuration modes, insufficient bit width and during initialization, this pin will not be used as normal user I/O, for high resistance or weak pull-up. (3) Once configured, it can be used as user I/O.
D[7:4]	VCCIOL5	Multiplexing	Input/Output	Multiplexed configuration data pins: (1) Master SPI X8 mode, connected to the second flash in the same way as D[3:0]; (2) For other configuration modes and during initialization, this pin will not be used as a normal user I/O with a high resistance state or weak pull-up. (3) Once configured, it can be used as user I/O.

D[3:2]	VCCIOL5	Multiplexing	Input/Output	Multiplexed configuration data pins (1) Master SPI mode, X4/X8, is connected to the data output of the SPI flash as data input. "D2" is connected to the 3rd output of the 4/8-bit data of the external SPI flash (e.g. DQ2, W#, WP#, IO2), "D3" is connected to the 4th output of the 4/8-bit data of the external SPI flash (e.g. DQ3, HOLD#, IO3, etc.), which requires a weak external pull-up resistor of 4.7KΩ to its corresponding VCCIO. (2) For Slave Parallel mode, as bits [3, 4] of the data bus. (3) For other configuration modes and during initialization, this pin will not be used as a normal user I/O with a high resistance state or weak pull-up. (4) Once configured, it can be used as user I/O.
MISO_D1_DI	VCCIOL5	Multiplexing	Input/Output	Multiplexing configuration data pins (1) Master SPI mode, X1, "MISO" is connected as data input to the data output of SPI flash (e.g. DQ1, Q, SO, IO1, etc.). (2) Master SPI X2, X4, X8 modes, "D1" is connected to the 2nd serial data output port of the SPI flash (e.g:DQ1, Q, SO, IO1, etc.). (3) For Slave Parallel mode, as bit [2] of the data bus. (4) For Slave Serial mode, "DI" is used as the data input pin. (5) In other configuration modes and during initialization, this pin is used as a normal user I/O with a high resistance state or weak pull-up. In other configuration modes such as JTAG, the state on this pin is ignored. (6) Once configured, it can be used as user I/O.
MOSI_D0	VCCIOL5	Multiplexing	Input/Output	Multiplexing of configuration data pins; (1) "MOSI", Master SPI X1 mode serial data output, is connected to the data input pins of the external SPI flash (e.g. DQ0, D, SI, IO0, etc.). After the command and address are sent to the external SPI flash, this pin outputs a high resistance state or a weak pull-up (depending on the state of the PULL_UP pin) (2) Master SPI X2/X4/X8 mode, bidirectional data port, used as command and address output to external SPI flash. lowest bit data input receives data from external SPI flash. Bidirectional data pins connected to external SPI flash (e.g. DQ0, D, SI, IO0, etc.) (3) "D0", for Slave Parallel mode, as bit [1] of the data bus, i.e. the lowest bit; (4) For other configuration modes and during initialization, this pin will not be used as a normal user I/O with a high resistance state or weak pull-up. (5) Once configured, it can be used as user I/O.

CSO_DOUT	VCCIOL5	Multiplexing	Output(OD)/Output	Multiplexed configuration pins. Only needed for cascading. (1) Master SPI, X1 mode, can be used as a cascade data output; otherwise, for other configuration modes and during initialization, this pin will not be used, as a normal user I/O, in a high resistance state or weak pull-up. (2) Slave Serial mode configuration for cascade data output; otherwise, for other configuration modes and during initialization, this pin is not used as a normal user I/O with a high resistance state or weak pull-up. (3) In any configuration mode, the bitstream debugging option (lowest security level) is checked and it can output data. (4) In Slave Parallel cascade mode configuration, it is available as an open-drain output of the chip select signal; it is connected to the CS_N pin of the downstream device and requires an external pull-up resistor of 330 Ω to be added to its corresponding VCCIO. Otherwise, for other configuration modes and during initialization, this pin will not be used as a normal user I/O, for high resistance or weak pull-up. (5) For other configuration modes and during initialization, this pin will not be used as a common user I/O, and it is in a high-impedance state or a weak pull-up.

Configuration level selection

The SCBV pin is used to select the Bank voltages where the multiplexed configuration pins are located. Use this pin to determine the voltage difference between the BANKCFG and the voltage of the BANK containing the multiplexed configuration pins, and the I/O voltage operating range of these Banks. This pin is always active for BANKCFG, but only for the Bank containing the multiplexed configuration pin during configuration.

When the voltage of VCCIOCFG is 2.5V or 3.3V, this pin must be connected to high level (can be directly connected to VCCIOCFG). When the voltage of VCCIOCFG is 1.8V or lower, this pin must be connected to low (can be connected directly to ground).

Different configuration modes will use different configuration multiplexed IOs, which are located on BANKL4 and BANKL5. The following table describes the corresponding settings of VCCIOCFG, VCCIOL4, VCCIOL5 and SCBV in different configuration modes.

Table 5 Power supply and SCBV connection in different configuration modes

CONFIGURATION MODE	CONFIGURATION INTERFACE IO LEVEL	RELATED BANK VOLTAGE			SCBV
		VCCIOCFG	VCCIOL4	VCCIOL5	
JTAG	3.3V	3.3V	Any	Any	VCCIOCFG
	2.5V	2.5V	Any	Any	VCCIOCFG
	1.8V	1.8V	1.8V/1.5V/1.2V	1.8V/1.5V/1.2V	GND
	1.5V	1.5V	1.8V/1.5V/1.2V	1.8V/1.5V/1.2V	GND
	1.2V	1.2V	1.8V/1.5V/1.2V	1.8V/1.5V/1.2V	GND
SLAVE SERIAL, SLAVE PARALLEL, AND MASTER SPI(X1/X2/X4)	3.3V	3.3V	Any	3.3V	VCCIOCFG
	2.5V	2.5V	Any	2.5V	VCCIOCFG
	1.8V	1.8V	1.8V/1.5V/1.2V	1.8V	GND
	1.5V	1.5V	1.8V/1.5V/1.2V	1.5V	GND
	1.2V	1.2V	1.8V/1.5V/1.2V	1.2V	GND
MASTER SPI(X8)	3.3V	3.3V	3.3V	3.3V	VCCIOCFG
	2.5V	2.5V	2.5V	2.5V	VCCIOCFG
	1.8V	1.8V	1.8V	1.8V	GND
	1.5V	1.5V	1.5V	1.5V	GND
	1.2V	1.2V	1.2V	1.2V	GND

This chapter mainly describes in detail the application interfaces for the four configuration modes supported by the Titan2 family of FPGAs listed in Table 1.

(1) JTAG configuration mode

The schematic diagram of the JTAG configuration mode application interface is shown in Figure 1.

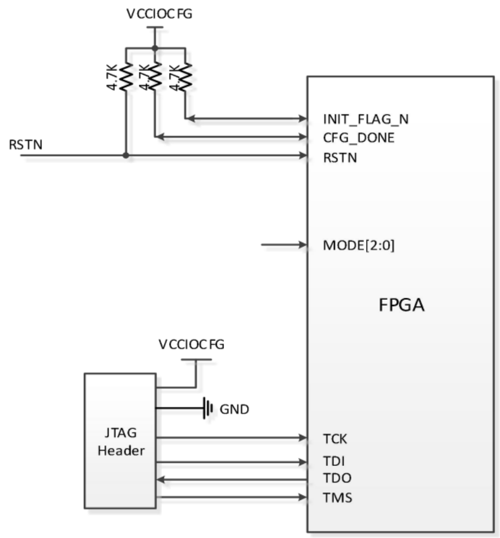


Figure 1 JTAG mode application schematic

A low level of RSTN will reset the configuration logic, it is recommended to use a 4.7K external resistor to pull up to VCCIOCFG. During initialization, if INIT_FLAG_N is connected low as input, it will cause the Titan2 FPGA to stay in the initialization phase, it is recommended to use a 4.7K external resistor to pull up to VCCIOCFG. CFG_DONE output high indicates that the chip is in user mode. a low external input will cause the Titan2 FPGA to stay in the configuration phase. it is recommended to use a 4.7K external resistor to pull up to VCCIOCFG.

TCK test clock in JTAG mode needs to be provided externally; the external can control the jump of the JTAG internal TAP state machine by changing the state of TMS, thus selecting the configuration bit stream write (TDI) or on-chip data readback (TDO). In addition to configuration programming, the JTAG interface is also commonly used for in-circuit debugging and boundary-scan testing. in JTAG mode, it is recommended that TDI, TCK, and TMS be pulled up to VCCIOCFG with 12-15K resistors to provide a stable initial input level.

The typical timing of the JTAG programming mode is shown in the following diagram.

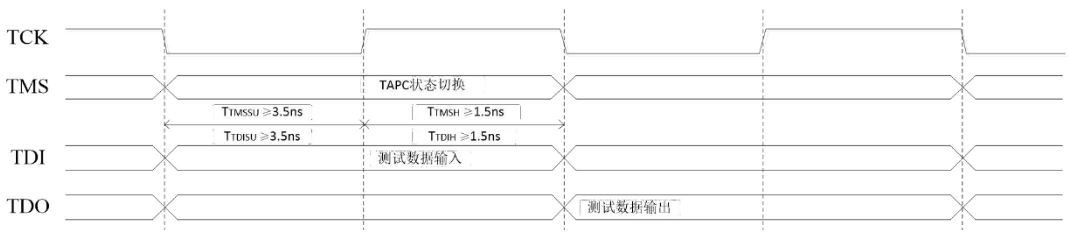


Figure 2 JTAG programming typical timing diagram

Since the JTAG interface can be used to program the eFuse, in order to prevent the eFuse from being incorrectly written due to interference signals caused by the environment on the JTAG interface, it is recommended that the user lock all the function bits of the eFuse after confirming the application scenario of the chip. See the eFuse introduction for details.

JTAG Cascade Mode

Multiple devices can be daisy-chained with JTAG, and the connections are shown in the figure below.

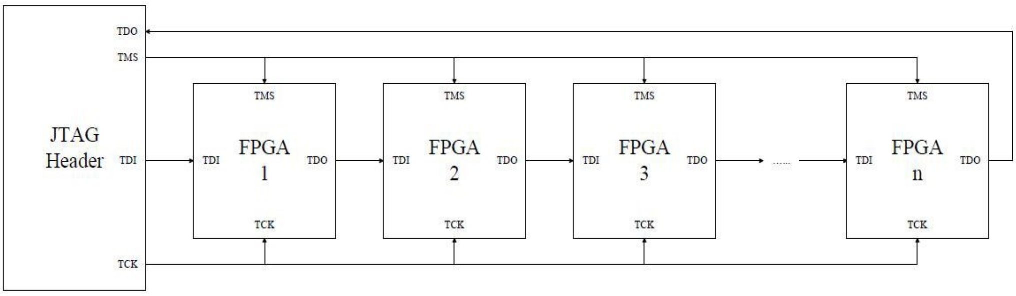


Figure 3 JTAG cascade application schematic

In this way, the mode pin should be set to JTAG mode. After connecting the download cable, the PDS software on the host will scan all the devices in the JTAG chain, and the user can choose to program the corresponding FPGAs for download.

The TCK and TMS signals connect all devices in the JTAG chain, so the signal quality of both affects the maximum frequency and reliability of the JTAG configuration. Note that when cascading JTAG, only operations on the external Flash of the first FPGA stage are supported.

(II) Master SPI configuration mode

The Master SPI configuration interface is shown below:

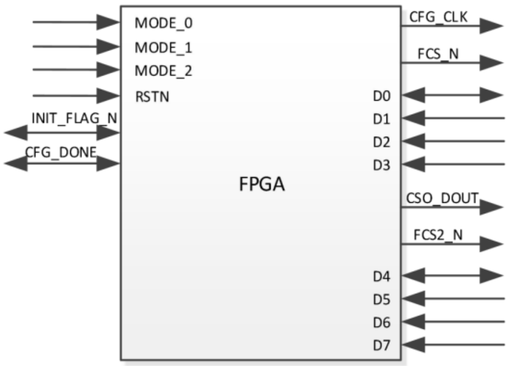


Figure 4 Master SPI Configuration Interface

The Master SPI configuration interface signals are described in the following table.

Table 6 Master SPI Interface Signal Descriptions

NAME	I/O	DEDICATED / MULTIPLEXED	DESCRIPTION
RSTN	I	Dedicated	Asynchronous full chip reset signal, low active
CFG_CLK	O	Dedicated	Configuring the clock
MODE[2:0]	I	Dedicated	Configuration mode pins
			001: Master SPI configuration mode (x1/x2/x4/x8)

INIT_FLAG_N	open a leak	Dedicated	Before sampling the MODE[2:0] pins, INIT_FLAG_N is an input that can be delayed by holding it low for configuration. After sampling the MODE[2:0] pins, INIT_FLAG_N is an open drain, indicating whether the configuration process is in error. 0: Error 1: Correct
FCS_N	O	Multiplexing	SPI Flash chip select signal, low active
FCS2_N	O	Multiplexing	x8 mode second SPI Flash chip select signal, low active
D[7:0]	IO	Multiplexing	Serial input/output data bus, CFG_CLK rising or falling edge sampling D[7:4] is the serial input and output data bus of the second SPI Flash in x8 mode
CSO_DOUT	O	Multiplexing	Daisy chain data output, CFG_CLK falling edge send
CFG_DONE	open a leak	Dedicated	Indicates configuration completion 0: FPGA not configured 1: FPGA is configured
SCBV	I	Dedicated	Configuration voltage selection 0: VCCIOCFG voltage is 1.8V or lower 1: The voltage of VCCIOCFG is 2.5V or 3.3V

In Master SPI mode, the bitstream is usually stored in the external SPI Flash. To select Master SPI mode, it is recommended to connect MODE[2:1] to ground via a pull-down resistor and MODE[0] to the VCCIOCFG power supply via a pull-up resistor. By powering up or giving RSTN a low pulse to start programming, the Titan2 series FPGA will actively read the bitstream from external Flash. The FPGA automatically uses fast read X1 mode, sampling data on the rising edge and fetching data from address 0 of the serial NOR FLASH. After fetching part of the mode setting instruction at the beginning of the bitstream, it switches to the mode specified by the instruction for bit width, frequency, sampling edge, etc. and continues to read the bitstream. After all the bit streams are removed from the Flash, FCS_N and FCS2_N (when using x8 bit width) are set to 1 and the Master SPI mode operation ends.

If the set CFG_CLK frequency is not higher than 15.38MHz, rising edge sampling can be used. If it is higher than 15.38MHz, it should be set to use falling edge sampling.

CFG_CLK is internally generated and output by the chip; the monitoring pins INIT_FLAG_N and CFG_DONE are used to determine whether the programming is finished; Master SPI supports three data bit width modes of 1/2/4/8bit, the FPGA will select the data bit width by parsing the bit stream, and for different data bit widths, the data bit selection, data input/output direction will be different, as described below:

When the X1 bit is wide, the FPGA pin D[0] is connected to the data input of SPI Flash as command output and pin D[1] is connected to the data output of SPI Flash as data input. The application schematic is shown below:

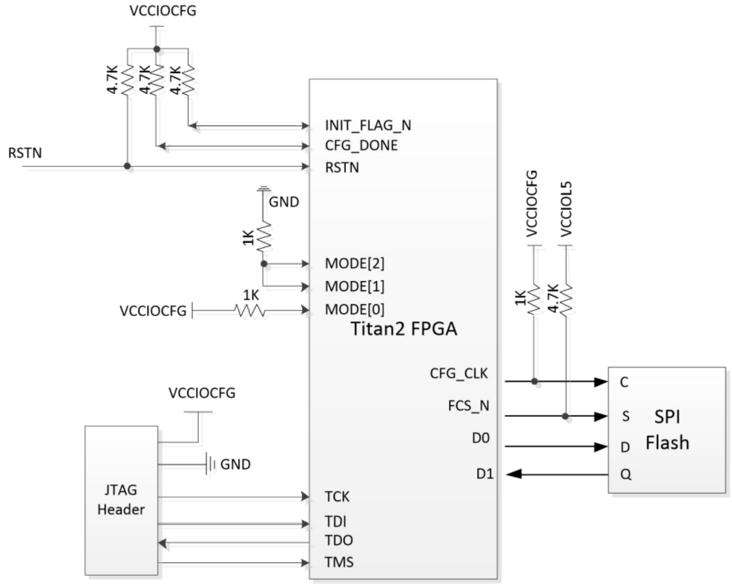


Figure 5 Master SPI Configuration Mode X1 Application Schematic

At X2 bit width, pin D[1:0] is the data bus. The application schematic is shown below:

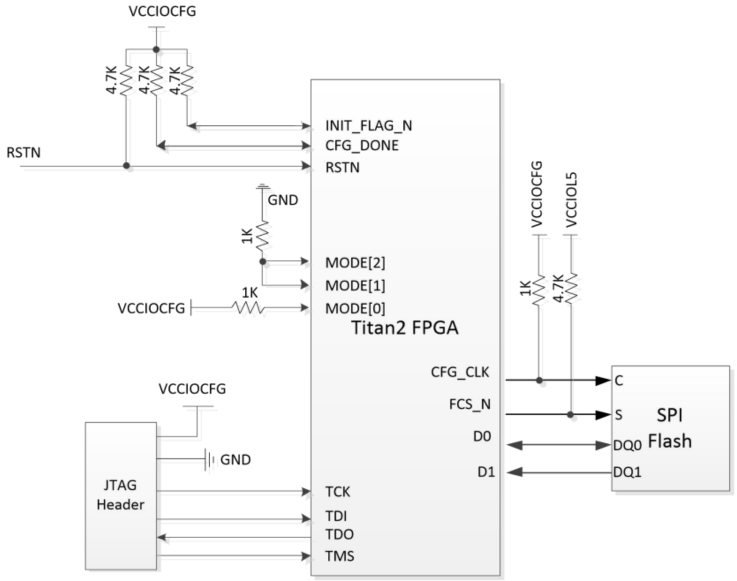


Figure 6 Master SPI Configuration Mode X2 Application Schematic

At X4 bit width, pin D[3:0] is the data bus. The application schematic is shown below: DQ[3:2] must have pull-up resistors.

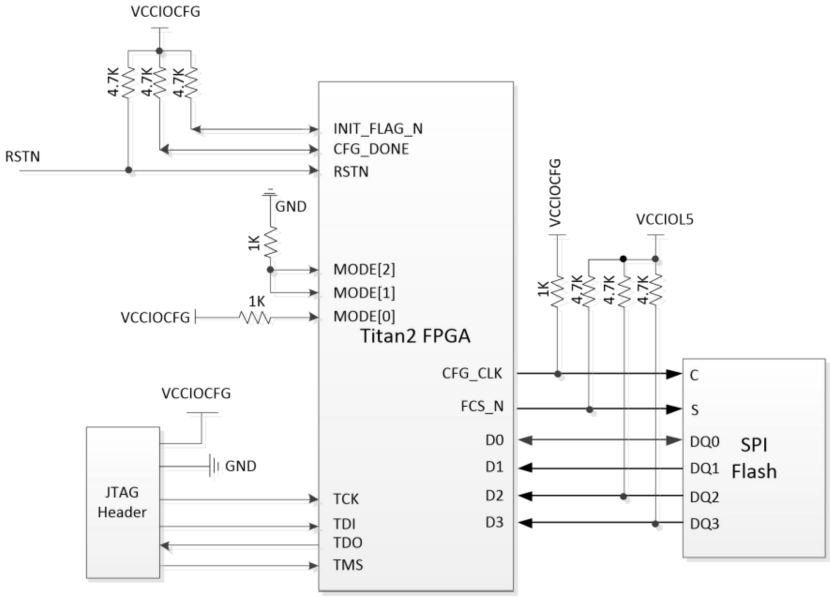


Figure 7 Master SPI Configuration Mode X4 Application Schematic

At X8 bit width, pin D[3:0] is the data bus of the first SPI Flash and pin D[7:4] is the data bus of the second SPI Flash. The application schematic is shown below:

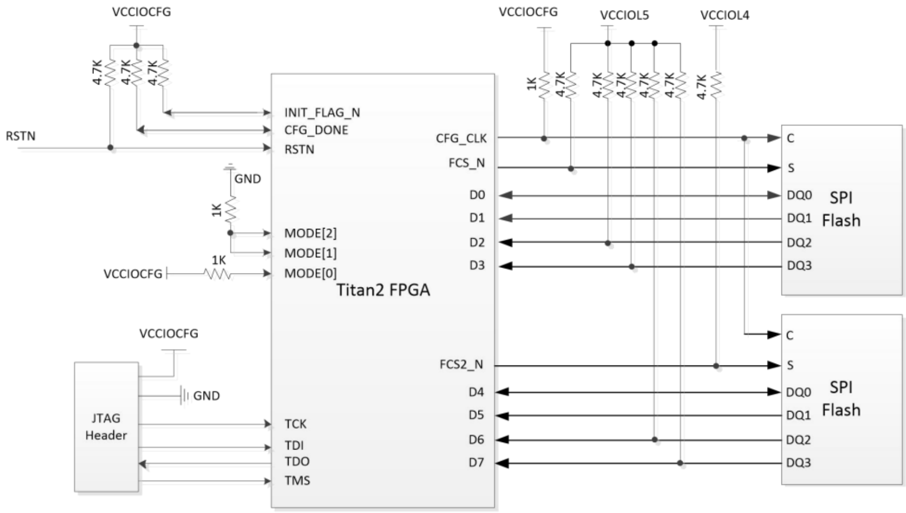


Figure 8 Master SPI Configuration Mode X8 Application Schematic

The typical timing of the Master SPI programming mode is shown in the following diagram:

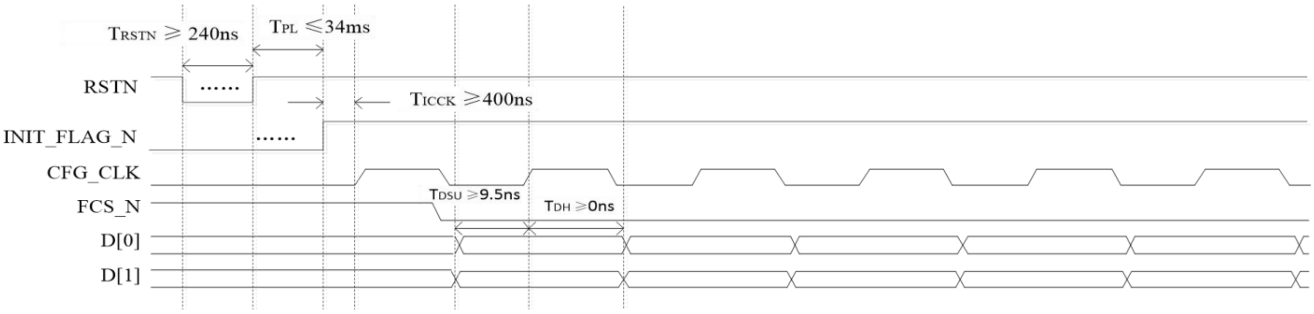


Figure 9 1bit Master SPI Programming Typical Timing Diagram

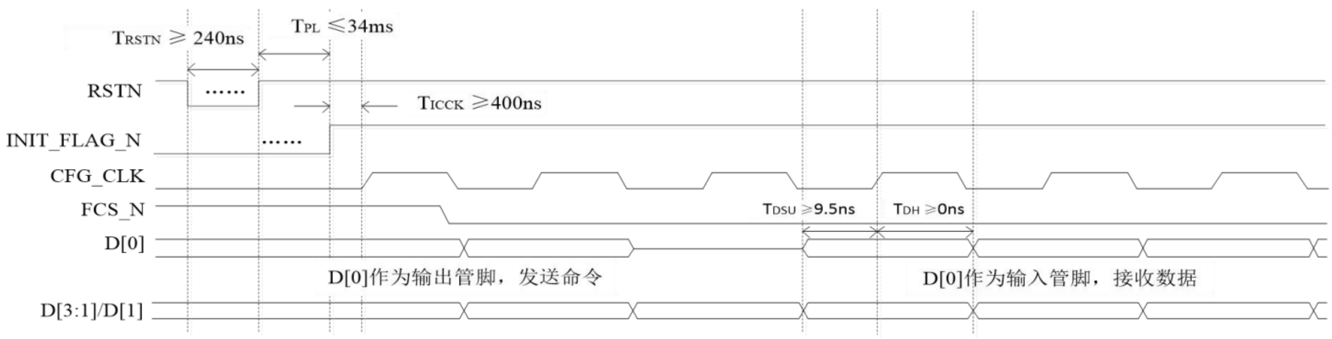


Figure 10 2/4bit Master SPI Programming Typical Timing Diagram

Some of the SPI Flash models supported by Titan2 series FPGAs are shown in the table below.

Table 7 Supported SPI Flash Models

MODEL	MANUFACTURER	CAPACITY
AT45DB081E	ADESTO	8Mb
S25FL64	CYPRESS	64Mb
S25FL128	CYPRESS	128Mb
S25FL256	CYPRESS	256Mb
S25FL512	CYPRESS	512Mb
EN25Q80C	ESMT	8Mb
FH25VQ80D	FENTECH	8Mb
M25P16	Micron	16Mb
M25P32	Micron	32Mb
M25P64	Micron	64Mb
M25P128	Micron	128Mb
N25Q32	Micron	32Mb
N25Q64	Micron	64Mb
N25Q128	Micron	128Mb
N25Q256	Micron	256Mb
N25Q512	Micron	512Mb
MD25Q80C	GigaDevice	8Mb
GD25Q80C	GigaDevice	8Mb
GD25Q32C	GigaDevice	32Mb
GD25Q64C	GigaDevice	64Mb
GD25Q128C	GigaDevice	128Mb
GD25Q256D	GigaDevice	256Mb
GD25Q512C	GigaDevice	512Mb
XM25QH16B	Wuhan New Core	16Mb
XM25QH32B	Wuhan New Core	32Mb
XM25QH64A	Wuhan New Core	64Mb
XM25QH128A	Wuhan New Core	128Mb
XM25QH256B	Wuhan New Core	256Mb
W25Q40CL	WINBOND	4Mb
W25Q80	WINBOND	8Mb
W25Q16	WINBOND	16Mb

W25Q32	WINBOND	32Mb
W25Q64Q	WINBOND	64Mb
W25Q64M	WINBOND	64Mb
W25Q128Q	WINBOND	128Mb
W25Q128M	WINBOND	128Mb
W25Q256	WINBOND	256Mb
W25Q512	WINBOND	512Mb
IS25LP016D	ISSI	16Mb
IS25LP032D	ISSI	32Mb
IS25LP064D	ISSI	64Mb
IS25LP128D	ISSI	128Mb
IS25LP256D	ISSI	256Mb
IS25LP512D	ISSI	512Mb
IS25WP016D	ISSI	16Mb
IS25WP032D	ISSI	32Mb
IS25WP064D	ISSI	64Mb
IS25WP128D	ISSI	128Mb
IS25WP256D	ISSI	256Mb
IS25WP512D	ISSI	512Mb
MX25L32	MXIC	32Mb
MX25L64	MXIC	64Mb
MX25L128	MXIC	128Mb
MX25L256	MXIC	256Mb
MX25L512	MXIC	512Mb
ZB25VQ16	ZBIT	16Mb
ZB25VQ32	ZBIT	32Mb
SM25QH256M	SSMEC	256Mb

Note: The selection of Flash should be considered according to the actual storage space requirements (such as the actual bitstream size, see Table 11 for the common bitstream size of Titan2 devices), i.e., if using a common bitstream, select a Flash larger than 128 Mb. The above Flash is the model already supported by PDS.

Master SPI Serial Daisy Chain

Master SPI configuration mode supports SPI serial daisy-chain configuration using Master SPI mode and Slave Serial mode.The first device in the SPI serial daisy-chain uses Master SPI mode and the other devices use Slave Serial mode.SPI serial daisy-chain configuration starts with the lowest-level device and ends with The first device is configured last. The logic block diagram is shown below:

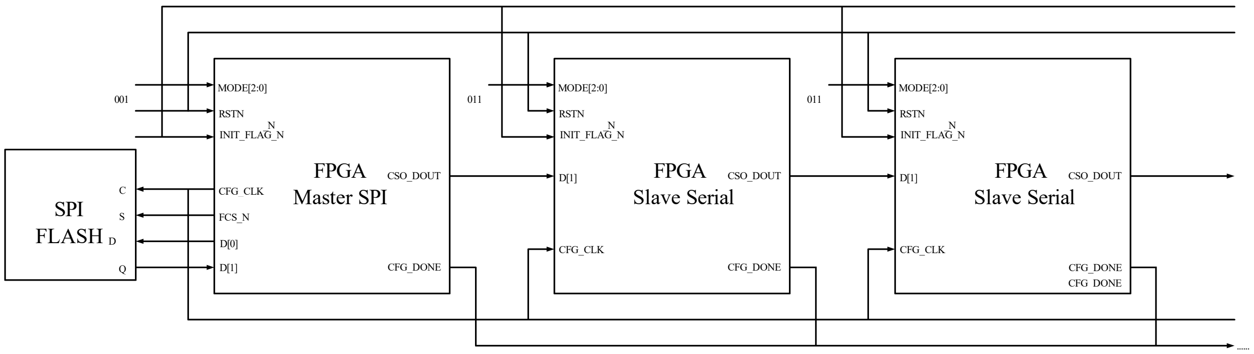


Figure 11 Master SPI Serial Daisy Chain Application Schematic

(III) Slave Serial configuration mode

The Slave Serial configuration interface is shown below:



Figure 12 Slave Serial Configuration Interface

The Slave Serial configuration interface signals are described in the following table.

Table 8 Slave Serial Configuration Interface Signal Description

NAME	I/O	DEDICATED / MULTIPLEXED	DESCRIPTION
RSTN	I	Dedicated	Asynchronous full chip reset signal, low active
CFG_CLK	I	Dedicated	Configuring the clock
MODE[2:0]	I	Dedicated	Configuration Mode Pins 111: Slave Serial mode
INIT_FLAG_N	open drain	Dedicated	Before sampling the MODE[2:0] pins, INIT_FLAG_N is an input that can be delayed for configuration by holding it low. After sampling the MODE[2:0] pins, INIT_FLAG_N is an open drain, indicating whether the configuration process is in error. 0: Error 1: Correct
CFG_DONE	open drain	Dedicated	Indicates configuration completion 0: FPGA not configured 1: FPGA is configured
DI	I	Multiplexing	The serial input data bus, the rising edge sampling of CFG_CLK is used for its own configuration, and the falling edge sampling is used for the configuration of the next stage when cascading.
CSO_DOUT	O	Multiplexing	Daisy chain data output, CFG_CLK falling edge send
SCBV	I	Dedicated	Configuration voltage selection 0: VCCIOCFG voltage is 1.8V or lower 1: The voltage of VCCIOCFG is 2.5V or 3.3V

To select the Slave Serial mode, it is recommended to connect MODE[2:0] to VCCIOCFG through a pull-up resistor. The application schematic of the Slave Serial configuration mode is shown below:

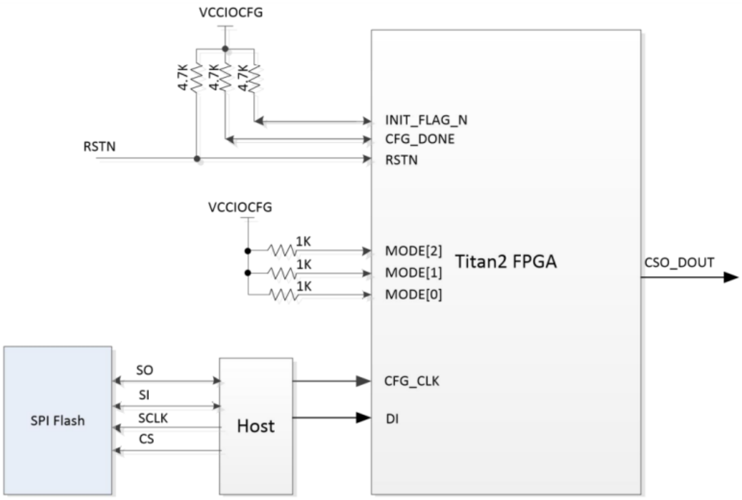


Figure 13 Slave Serial mode application schematic

In Slave Serial configuration mode, the power-up and data loading of multiple chips on the board can be controlled by a Host, which can be a microprocessor, CPLD or other FPGA. DONE to determine if the programming is finished.

When the Host sends a bitstream to the Titan2 device without enabling Wait PLL Lock (not enabled by default), the device will release control of CFG_DONE sometime between 100 no-op type 1 packet headers at the end of the bitstream, at which point CFG_DONE will be pulled high due to the external pull-up resistor. The portion of the bitstream after CFG_DONE is pulled high is used to provide the clock for device wake-up, so it is necessary to ensure that the bitstream is sent intact before terminating the clock. If wait is enabled, the clock cannot be terminated until CFG_DONE is pulled high, and at least 100 more clocks need to be given after CFG_DONE is pulled high.

When using serial daisy-chain configuration, each level of the device will use CFG_CLK rising edge sample DI for its own configuration in addition to falling edge sample DI for output configuration of the next level. Therefore, when cascading configuration, Host needs to consider both rising and falling edges to meet the establishment hold time requirements.

Typical timing for Slave Serial programming mode is shown in the following diagram:

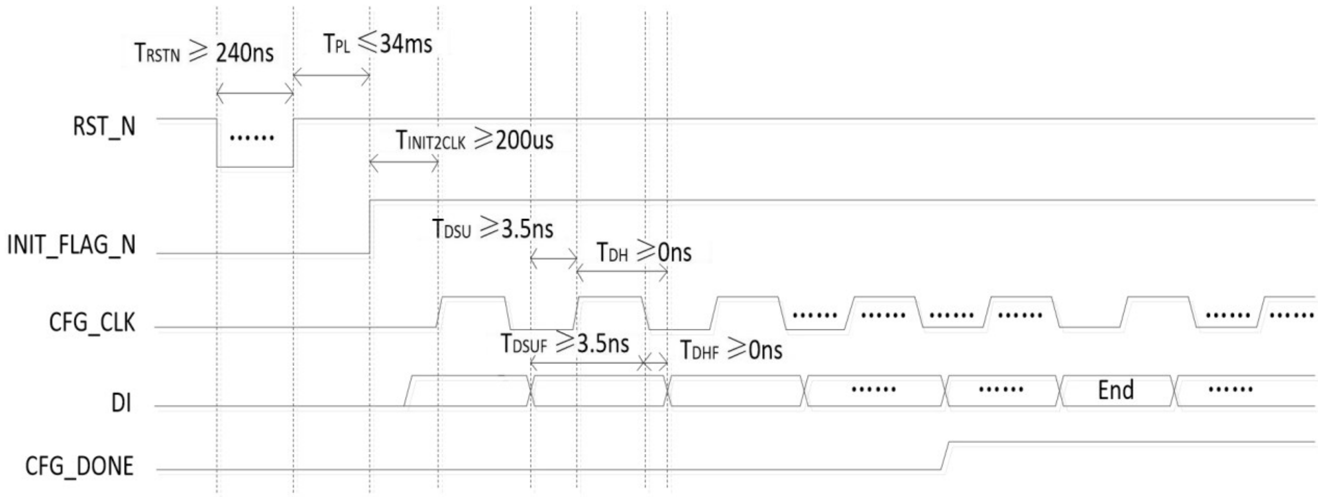


Figure 14 Slave Serial Programming Typical Timing Diagram

Slave Serial Daisy Chain

The typical application of Slave Serial configuration mode is for serial daisy chaining. All devices can be configured in serial cascade using Slave Serial mode in its entirety. When cascading configuration, the configuration starts with the last device (furthest away) and ends with the first device. Cascading bitstreams for cascade configuration requires the use of the Fabric Configuration tool to combine several bitstreams into one cascaded bitstream.

The serial cascade can control the configuration of multiple cascade chips on the board through a single master chip (Host). The connections are shown in the diagram below.

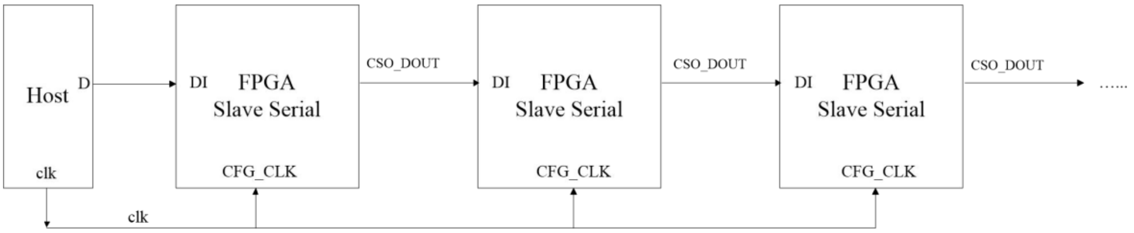


Figure 15 Slave Serial Configuration Mode Serial Daisy Chain Connection

When using serial daisy chains, care needs to be taken to:

Meet the first level timing

As shown in Figure 14, the timing of Host send needs to meet the CFG_CLK rising edge and falling edge sample DI establishment hold time requirements.

Meet the post timing

The first stage timing can be adjusted by Host to meet the output, while the post-stage input are from the front stage, can be adjusted mainly PCB alignment and other introduced delays. As shown in the figure below, the delay parameters $T_{CLKDELAY}$ and $T_{DOUTDELAY}$ need to be adjusted to meet the requirements of the post-stage devices along the two establishment hold time.

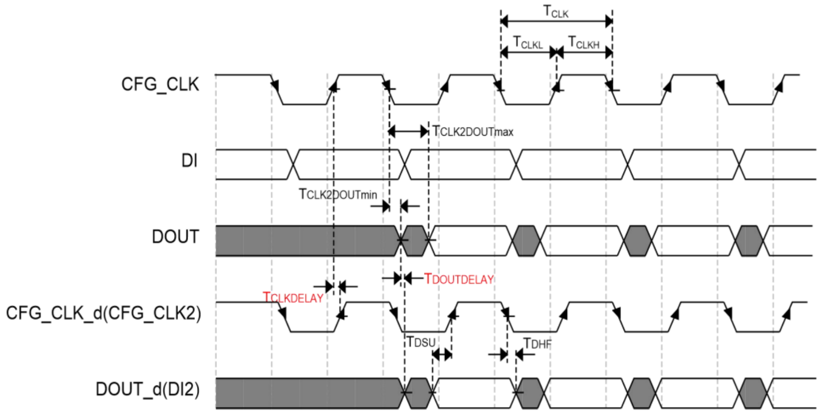


Figure 16 Adjustment Timing Example - Rear Stage

From the above figure, we can get that if the falling edge transmits data, if the rising edge establishment time is satisfied, then the falling edge establishment time must be satisfied, and if the falling edge hold time is satisfied, then the rising edge hold time must be satisfied. Therefore, $T_{CLKDELAY}$ and $T_{DOUTDELAY}$ need to meet the following formula (unit: ns)

$$T_{DSU} = (T_{CLK} + T_{CLKDELAY}) - (T_{CLKH} + T_{CLK2DOUTmax} + T_{DOUTDELAY}) \geq 3.5$$

$$T_{DHF} = T_{CLK2DOUTmin} + T_{DOUTDELAY} - T_{CLKDELAY} \geq 0$$

Where: $10.5 - T_{CLKL} \leq T_{CLKDELAY} - T_{DOUTDELAY} \leq 2$

(IV) Slave Parallel configuration mode

The Slave Parallel configuration interface is shown below:



Figure 17 Slave Parallel Configuration Interface

The Slave Parallel configuration interface signals are described in the following table.

Table 9 Slave Parallel Configuration Interface Signal Description

NAME	I/O	DEDICATED / MULTIPLEXED	DESCRIPTION
RSTN	I	Dedicated	Asynchronous full chip reset signal, low active
CFG_CLK	I	Dedicated	Configuring the clock
MODE[2:0]	I	Dedicated	Configuration Mode Pins 110: Slave Parallel Mode
INIT_FLAG_N	open drain	Dedicated	Before sampling the MODE[2:0] pins, INIT_FLAG_N is an input that can be delayed for configuration by holding it low. After sampling the MODE[2:0] pins, INIT_FLAG_N is an open drain, indicating whether the configuration process is in error. 0: Error 1: Correct
CFG_DONE	open drain	Dedicated	Indicates configuration completion 0: FPGA not configured 1: FPGA is configured
CS_N	I	Multiplexing	External parallel interface chip select signal, valid low. sampled on rising edge of CFG_CLK.(1)
RWSEL	I	Multiplexing	External parallel interface read/write control signal. sampled on rising edge of CFG_CLK. 0: Write 1: Read
D[31:0]	IO	Multiplexing	Parallel data bus, CFG_CLK rising edge sampling and sending
CSO_DOUT	O	Multiplexing	Daisy chain data output, CFG_CLK falling edge send
SCBV	I	Dedicated	Configuration voltage selection 0: VCCIOCFG voltage is 1.8V or lower 1: The voltage of VCCIOCFG is 2.5V or 3.3V

Note: (1) CS_N is a synchronization signal. After INIT_FLAG_N is pulled high, before configuring the chip, you need to keep CS_N high and give CFG_CLK at least 8 clock cycles to ensure the interface is in the initial working state.

In Slave Parallel mode, the power-up and data addition of multiple chips on the board can be controlled by a single Host chip.

To select Slave Parallel mode, it is recommended to connect MODE[0] to ground through a pull-down resistor, and connect MODE[2:1] to VCCIOCFG power supply through a pull-up resistor. The application diagram is as follows:

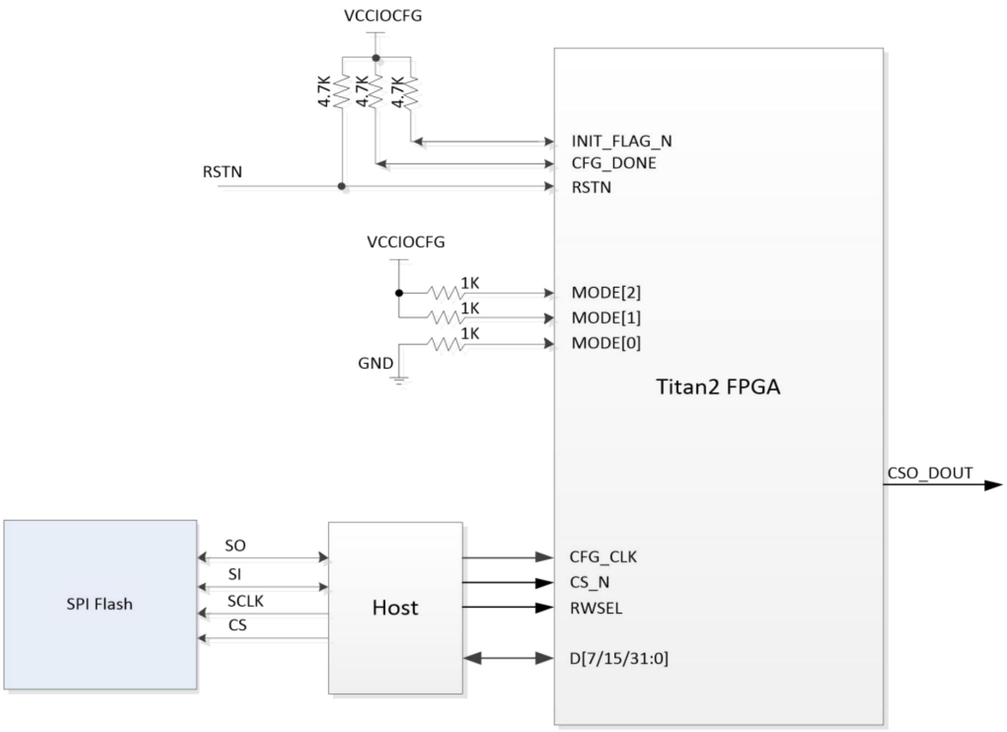


Figure 18 Slave Parallel Mode Application Schematic

This mode allows programming to be initiated by powering up or by giving a low pulse to the RSTN;

The end of programming is determined by monitoring pins INIT_FLAG_N and CFG_DONE; when Host sends a bitstream to the Titan2 device, if Wait PLL Lock is not enabled (not enabled by default), the device will release control of CFG_DONE sometime between the 100 no-op type 1 packet headers at the end of the bitstream, and due to the external pull-up resistor CFG_DONE will be pulled high at this point due to the presence of an external pull-up resistor. If wait is enabled, the clock cannot be terminated before CFG_DONE is pulled high, and CS_N is pulled high after CFG_DONE is pulled high. at least 100 more clocks need to be given after CS_N is pulled high.

Slave Parallel mode supports three data bit width modes of 8/16/32bit, and the device will select the data bit width by parsing the bit stream. For details, see the introduction of Bus Bit Width Auto Detection in Chapter 3.

The Titan2 device Slave Parallel interface supports several bit widths of x8/x16/x32. When using the Slave Parallel interface for configuration, you should pay attention to the bit sequence correspondence between data bus and data, refer to the bit sequence under different interface bit widths in Chapter 3.

A typical timing diagram for the Slave Parallel configuration interface is shown in the following figure.

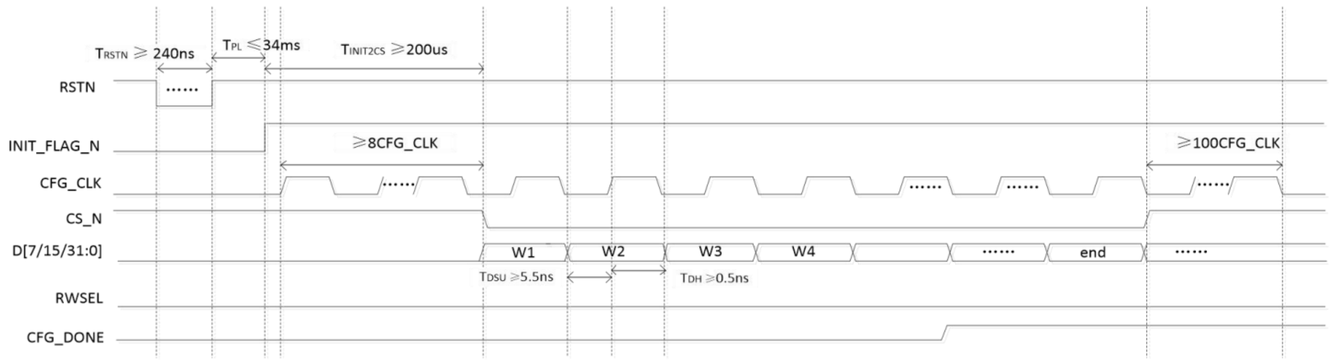


Figure 19 Slave Parallel Configuration Timing Diagram

When Host cannot send data streams continuously, non-continuous loading can be achieved by controlling CS_N or CFG_CLK of the Slave Parallel configuration interface, as shown in the following figure.

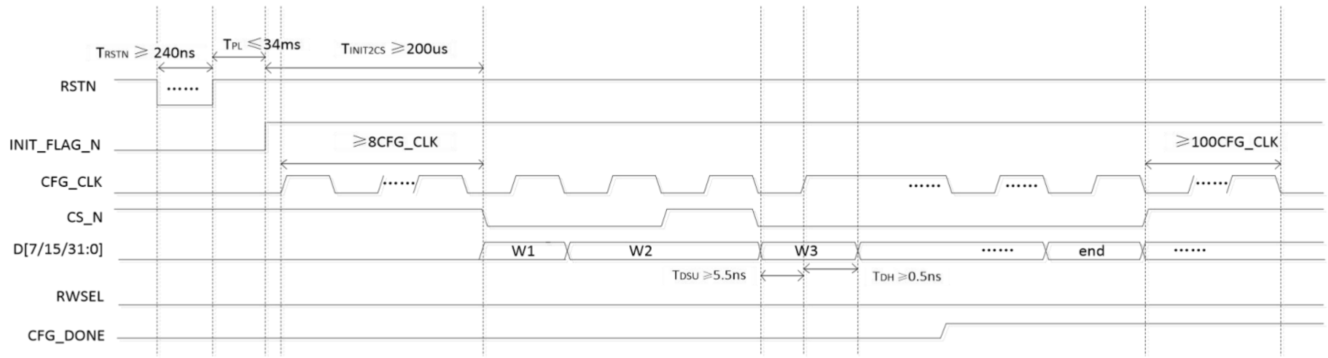


Figure 20 Slave Parallel Discontinuous Configuration Timing Diagram

After W2 is sent, the data loading is suspended by pulling up CS_N. Special attention needs to be paid. When pulling up CS_N to suspend data loading, the data needs to be kept unchanged, that is, D[31:0] remains unchanged when CS_N is high, otherwise It may lead to loading failure; after sending W3, by stopping the flip of CFG_CLK, the data loading can also be suspended. In this case, it is not necessary to keep D[31:0] unchanged.

In addition, the Slave Parallel interface also supports readback, the typical timing is shown in the figure below, refer to Slave Parallel interface readback for the specific readback process.

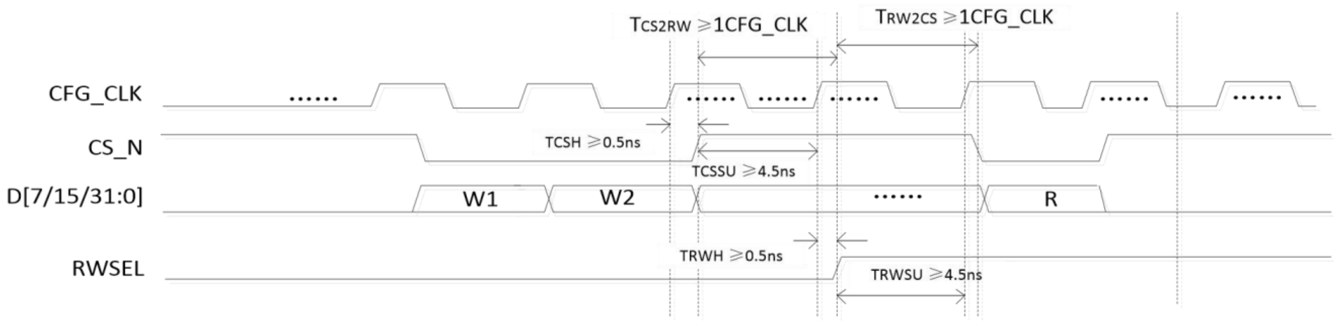


Figure 21 Slave Parallel Readback Timing Diagram

Slave Parallel Daisy Chain

A typical application of Slave Parallel configuration mode is for parallel daisy chain. All devices can use Slave Parallel mode for parallel cascade configuration. When cascading configuration, start configuration from the last-level device (the farthest end), and configure the first device last. The cascaded bitstream of the cascaded configuration requires the use of the Fabric Configuration tool to combine several bitstreams into a cascaded bitstream. (Use the Fabric Configuration tool to generate the location: Operations->Generate Daisy Chain File)

Parallel cascading can control the configuration of multiple cascading chips on the board through a main control chip (Host). The connections are shown in the figure below.

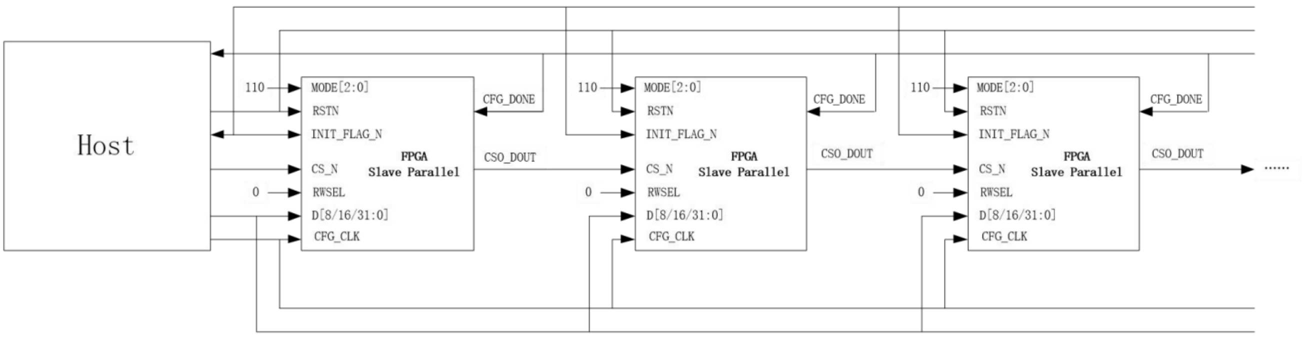


Figure 22 Slave Parallel Configuration Mode Parallel Daisy Chain Connection

III. Download configuration

(I) Configuration process

The download configuration process for Titan2 series FPGAs is shown in the following diagram:

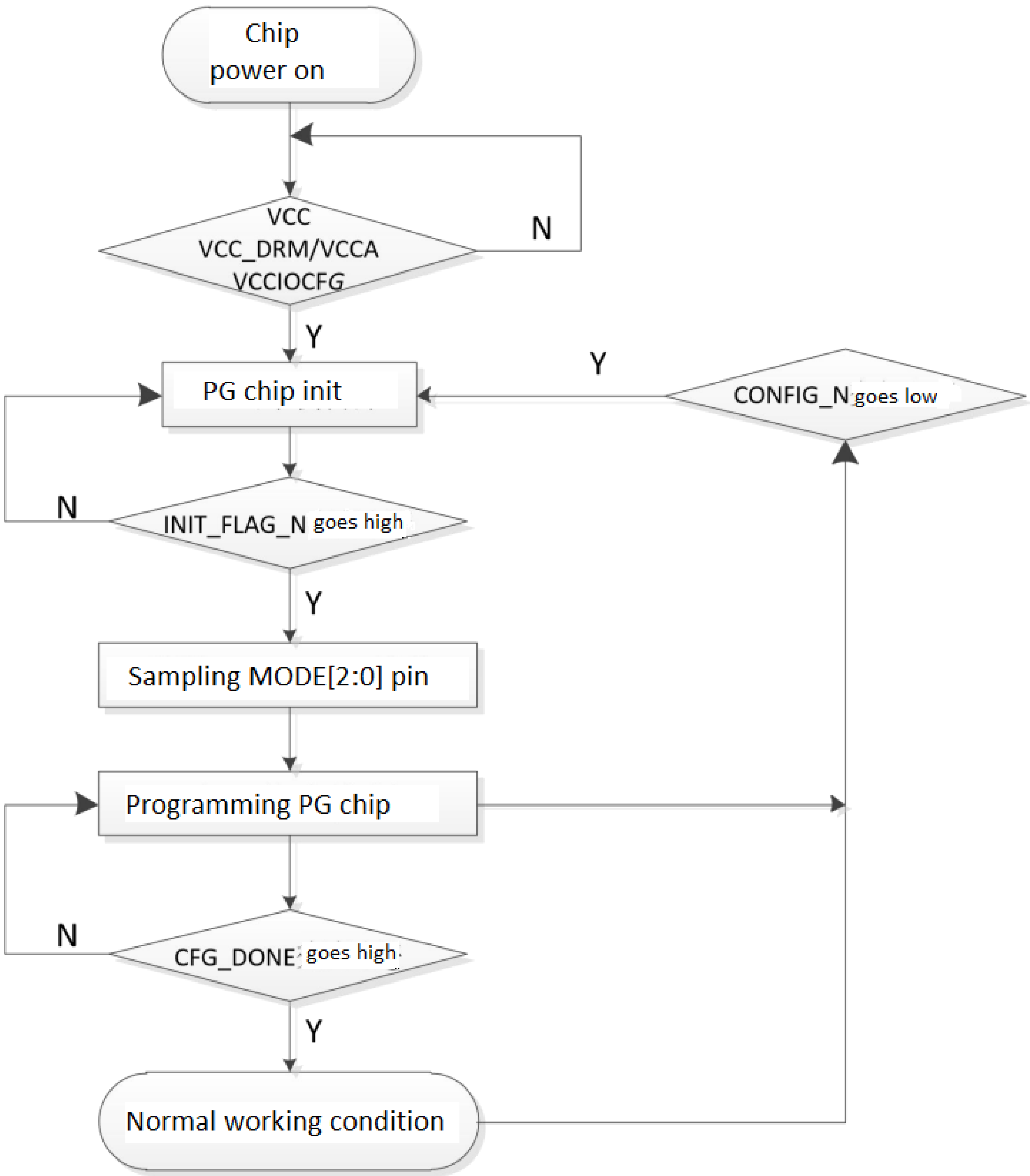


Figure 23 Download configuration flowchart

For all configured interfaces, the basic configuration steps are the same, including establishment, bitstream loading, and wake-up.

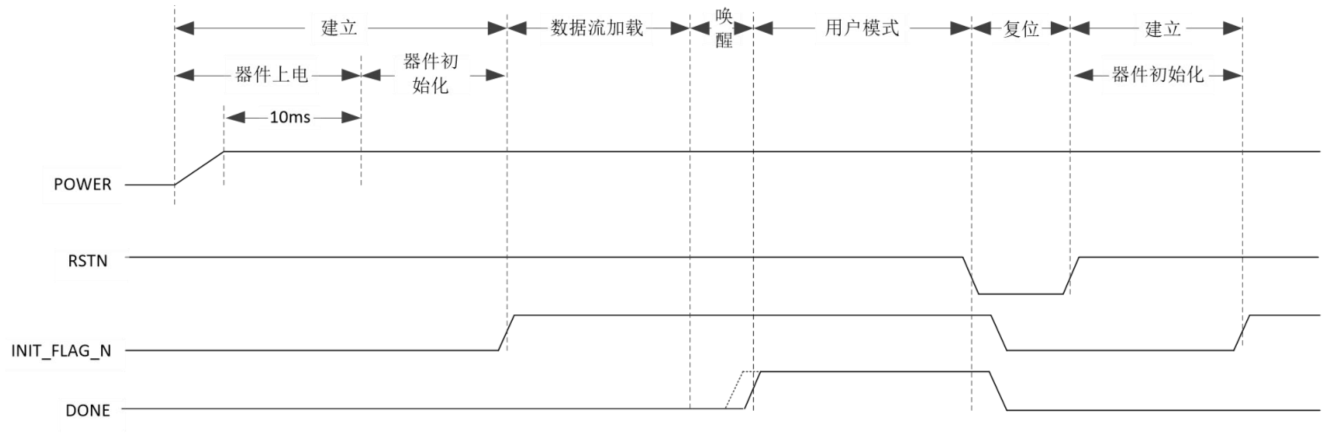


Figure 24 Download Configuration Timing

The configuration process of the Titan2 FPGA consists of the following stages.

1. Establishment

Establish and complete the operation of FPGA device initialization and configuration mode determination, including device power-on, device initialization and configuration mode selection.

1) Device Power-Up

After the device is powered on, the power-on reset circuit part of the configuration control system begins to work, and other circuit parts are still in the reset state. The power-on reset circuit uses the configuration control system master clock CLK generated by the device's internal crystal to count, and after counting 10ms, it means that the clock generated by the internal crystal has been stabilized, at which time the configuration control system power-on reset is released and the configuration control system starts to work.

2) Device initialization

After the device is powered on, the configuration control system is reset and released, and the configuration control system starts working, the device initialization operation is performed first.

Device initialization completes two operations: clearing the configuration memory and reading the eFuse contents. After initialization is complete, the flag signal INIT_COMPLETE goes high.

The configuration control system can be maintained to continue in the device initialization process by keeping the external INIT_FLAG_N pin low.

The configuration control system re-enters the device initialization process when one of the functions Reset, Hot Start, Version Fallback is triggered.

3) Configuration mode selection

When the INIT_FLAG_N signal is high, the configuration mode pin MODE[2:0] is sampled to select the configuration mode of the device. After the mode sampling is complete, CCS selects the configuration function of the multiplex pin.

2. Bitstream loading

Different configuration modes have different configuration interfaces. After the configuration mode selection is completed, the pin corresponding to the corresponding mode is set as the

configuration pin. This stage includes bus bit width check, synchronization, device flag check, pre-load option setting, data loading, CRC, post-load option setting and other operations.

1) Automatic bus bit width detection

In Slave Parallel configuration mode, the data bus bit width is automatically detected by CCS, and the bit width is matched by detecting the bus bit width auto-detect sequence consisting of two words 000000AA 08100020.

The bus bit width auto-detection sequence is at the beginning of the bit stream, before the synchronization word. ccs detects only the lower 8 bits of the parallel data bus.

When CCS performs detection, it first detects the 0xAA byte in the lower eight bits of the data bus, and after detecting 0xAA, it continues to detect whether the next byte is 0x08 (x8), 0x10 (x16) or 0x20 (x32). If the next byte is not one of the three above, CCS re-detects 0xAA until the 0xAA + 0x08/0x10/0x20 sequence is detected. After that, CCS sets the relevant data pin to the configuration data pin, the data bus switches to the detected bit width and starts to look for the synchronization word.

8-bit bus

FF	Filled words
FF	
FF	
FF	
00	Bus bit width auto-detection sequence
00	
00	
AA	
08	
10	
00	
20	
FF	Filled words
FF	
FF	
FF	
...	

16-bit bus

FFFF	Filled words
FFFF	
0000	Bus bit width auto-detection sequence
00AA	
0810	
0020	
FFFF	Filled words
FFFF	
...	

32-bit bus

FFFFFFFF	Filled words
000000AA	Bus bit width auto-detection sequence
08100020	
FFFFFFFF	Filled words
...	

2) Synchronization

The synchronization word 0x01332D94 is used for 32-bit word boundary alignment. After synchronization, subsequent unframing operations can be performed.

The data is considered valid by CCS only when the correct synchronization word is received; the packet processor of CCS performs the unpacking operation; any data before the synchronization word is ignored, except for the bus bit-width auto-detection sequence.

3) Master mode read operation selection

The read operation mode (including bit width and sample edge) is selected during Master SPI configuration mode.

When the Master SPI configuration mode is selected, the device resends a read command to the SPI Flash after the selection is completed and readjusts the data bus bit width according to the read command.

4) Reset CRC

Reset the CRC register.

5) Device identification check

Check if the device identifier in the bitstream matches the hardware, if not, the INIT_COMPLETE signal becomes low, the ID error flag is stored in the status register, and the configuration process is jumped out.

6) Pre-load option settings

Specify the operating state of the device in the configuration register before loading the configuration data:

Watchdog setting

Decryption initial vector setting

Key selection

Decryption options

Set whether to keep the multiplexed configuration port to continue to be used as a configuration port after the configuration is complete

Wake-up clock selection

Wake-up Timing Setting

Whether to wait for PLL version fallback setting when waking up

Master mode clock frequency setting

7) Loading configuration data

Writes configuration data frames to configuration memory.

8) CRC check after loading

After the configuration data is loaded, perform a CRC check. The CRC value generated by the software is stored in the CRC register and compared with the CRC value calculated by the CCS. If the two values are consistent, it means that the CRC check is passed, and all configuration information is correctly written into the configuration memory. If the two values are inconsistent, it means that the CRC check is different, then the INIT_N signal becomes low, and the CRC error flag is stored in the status register.

9) Post-load option settings

After the configuration data is loaded, specify the operating state of the device in the configuration register:

Set whether to allow the user to logically shut down the OSC

External port security level setting (whether to allow readback and reassignment)

3. Awakening

After the bitstream is loaded and passes CRC, the FPGA enters the wake-up phase. The configuration system first enables the logic outputs of all functional modules inside the FPGA. CRC checksum again before wake-up, and then de-synchronization operation is performed, indicating that the configuration operation is finished and the subsequent operation needs to be resynchronized. Finally, the wake-up is completed and the corresponding global signals are released gradually.

1) Global logic enable

After the CRC check is passed after loading, the logic output of all function modules inside the chip is enabled.

2) Wake up start

After the global logic is enabled, the wake-up operation is initiated.

3) Pre-wake CRC check

Before the wake-up operation is executed, CRC checksum is performed. The CRC value generated by the software is stored in the CRC register and compared with the CRC value calculated by CCS. If the two values agree, it means the CRC check is passed and the wake-up prep work is completed. If the two values do not agree, it means the CRC check is not passed, then the INIT_FLAG_N signal becomes low and the CRC error flag is stored in the status register.

4) Go to Synchronization

Indicates that the configuration process is finished and subsequent operations need to be resynchronized.

5) Awakening

After the desynchronization is completed, the wake-up circuit starts to work according to the set timing, and gradually releases the corresponding global signal. The wake-up sequence can be set in PDS (Project->Project Setting->Generate Bitstream->Startup). The following will introduce each global

signal in the default order.

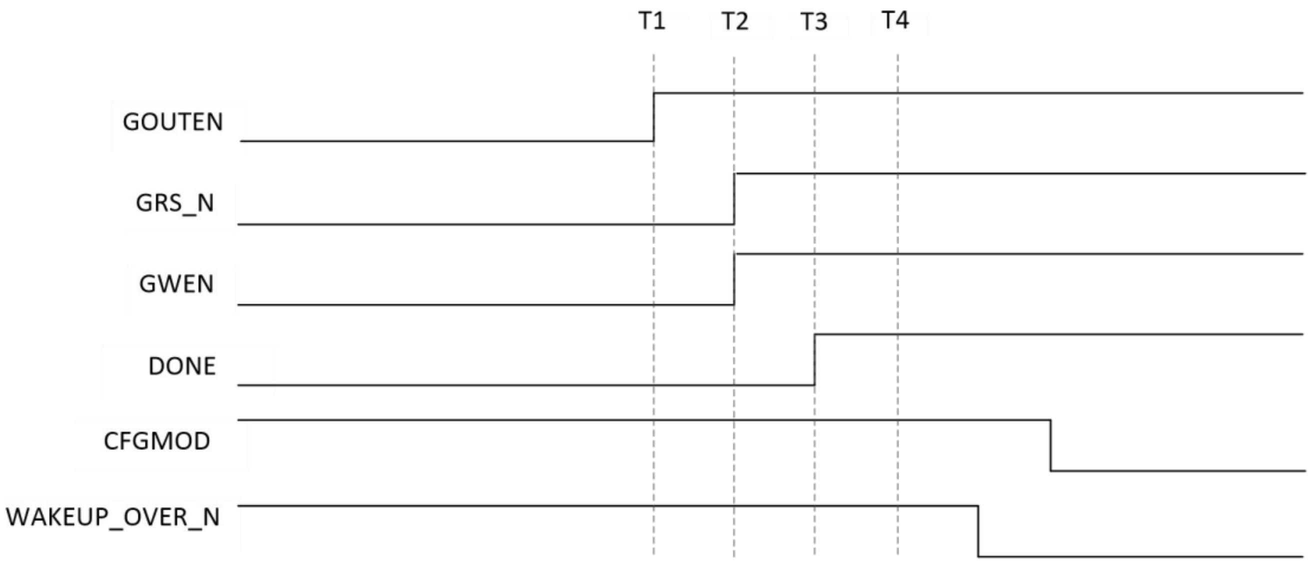


Figure 25 Wake-up Timing

The global logic enable signal GLOGEN is output to the fabric to enable each logic block; it is fed back to the CCS at the furthest end. the CCS receives the feedback signal GLOGEN_FB and enters the wake-up timing.

Before the configuration is completed, the global register reset signal GRS_N is low, which controls the GRS_N port of each module of the whole chip, so that the registers of each module are in the reset state; after the configuration is completed, the GRS_N signal of each module of the whole chip is released, so that the registers of each module enter the non-global reset state.

Until the configuration is completed, the global IO output enable signal GOUTEN is low and the outputs of all IOs are in high resistance. After the configuration is completed, the outputs of all IOs are enabled.

Table 10 GOUTEN Logic Functions

GOUTEN	0	1
IO used by the user	High-Z	User logic control
IO not used by the user	High-Z	Bitstream Settings(1)

Note: (1) The setting location in PDS tool is Project->Project Setting->Generate Bitstream->General->Unused IO Status, the default value is UNUSED, i.e. High-Z.

The global write enable signal GWEN is the global write enable for the internal memory resources of the FPGA chip. This signal is low until the configuration is completed, which disables the write operation of FPGA memory resources.

The multiplexed pin control signal CFGMOD releases the multiplexed pins used in the configuration process for user use. The wake-up completion indication signal WAKEUP_OVER_N indicates the successful completion of the wake-up process.

After the wake-up is finished, the FPGA enters user mode from configuration mode.

Note: Under the default wake-up timing, DONE is released after GOUTEN, GRS_N and GWEN to ensure that IO is already under user control when DONE is pulled high, but IO can be output after GOUTEN is released, i.e. IO is not high resistance before DONE. If you want the IO to be high resistance

before DONE, you can set DONE to be released before GOUTEN or at the same time. The storage resources controlled by GRS_N and GWEN, such as FF, are often used to drive IO output. Setting these two signals to be released before GOUTEN can ensure that the storage resource output is stable before driving the IO output, reducing the impact of the initial value of the storage resource. **That is, it is recommended to set the wake-up sequence as follows: DONE is released at T1, GWEN and GRS_N are released at T2, and GOUTEN is released at T3.**

If SCBV is connected to GND, i.e. VCCIOCFG supply is 1.8V or below, the initial value of storage resources such as FF will still be output after GOUTEN is released despite GWEN and GRS_N are released in advance. If the IO outputs abnormal pulses during the wake-up phase, it is necessary to check the driving logic of IO and modify the logic code to change the initial value of the driven storage resources such as FF to avoid the impact of the initial value.

If the user needs to reload the bitstream, the RSTN can be pulled down for a period of time and then released, and the FPGA will repeat the appeal process.

Delayed wake-up

The global wake-up signals GOUTEN, GRS_N, GWEN and DONE are user configurable and can be set to 1 at T1, T2, T3 or T4.

The number of cycles between two moments can be assigned by the user.

The CFG_DONE pin is set to 1 to release the drive control of this pin; CFG_DONE pin is an open-drain output pin, usually with external pull-up resistor. If the CFG_DONE pin does not become high (external drive is low) after the DONE signal becomes high, the wake-up process is suspended until the CFG_DONE pin becomes high, and then the subsequent wake-up process will be completed.

After DONE is set to 1, it needs to wait for the external signal CFG_DONE, and the external CFG_DONE driver is high before subsequent operations can be performed. Signals that are set to 1 after DONE is set to 1 need to wait for CFG_DONE to be high. Signals that are set to 1 before DONE is set to 1 and at the same time as DONE are not affected by the external CFG_DONE driver.

For example, if GOUTEN is set to 1 at T1, DONE is set to 1 at T2, GWEN is set to 1 at T3, and GRSN is set to 1 at T4, then GOUTEN and DONE are set to 1 according to the set timing sequence, and at T3, if CFG_DONE is externally driven to 1, then GWEN is set to 1, otherwise GWEN delays one time slot (the number of cycles between the two times can be matched) and continues to detect CFG_DONE at the next time slot. If CFG_DONE is high, GWEN is set to 1, otherwise GWEN delays for one time slot.

The delayed wake-up related settings are located in PDS under Project->Project Setting->Generate Bitstream->Startup.

4. IO Status Summary

The states of the different IOs throughout the process of the chip from initialization to entering user mode can be summarized in the following table:

Table 11 IO Status Summary

<table>
<tr><th colspan="2" rowspan="2">PROCESS</th><th rowspan="2">INIT</th><th rowspan="2">LOADING</th><th colspan="3">AWAKENING</th><th rowspan="2">USER MODE</th></tr>
<tr><th>GLOGEN AFTER</th><th>GOUTEN AFTER</th><th>CFGMOD AFTER</th></tr>
<tr><td colspan="2">General IO</td><td colspan="4">IO_STATUS_C(2)</td><td colspan="2">User(3)</td></tr>
<tr><td rowspan="2">Configuring</td><td>Enable(1)</td><td>IO_STATUS_C(2)</td><td colspan="4">IO_STATUS_C(2) & CCS(4)</td><td>User(3)</td></tr>
<tr><td>Non-enabling(1)</td><td colspan="5">IO_STATUS_C(2)</td><td>User(3)</td></tr>
</table>

Multiplexed IO	IO_STATUS_C(2)	Pull-up	User(3)
Configure dedicated IO	INIT_FLAG_N/RSTN/ CFG_DONE/TDO	Pull-up & CCS(4)	User(3)
	TCK/TDI/TMS/ MODE[2:0]	Pull-up	User(3)
	SCBV	High-Z	

Note: (1) Whether the configuration multiplex IO is enabled or not is determined by the MODE[2:0] setting, see Table 2 and Table 3 for details of the configuration pins.

(2) IO_STATUS_C controls whether the weak pull-up resistors of all user IOs (including multiplexed IOs) are enabled, as detailed in Table 4.

(3) "User" indicates that it is determined by user settings, such as IO constraints or software settings.

(4) "CCS" indicates that the relevant IO is controlled by the Configuration Control System (CCS) in this process, and the IO state depends on the CCS operating state.

(II) Configuration file

1. Bitstream Generation

The PDS design software generates configuration files with different extensions to accommodate different configuration scenarios, as described in the table below.

Table 12 PDS Configuration File Description

CONFIGURATION FILE EXTENSIONS	DESCRIPTION
.sbit	Binary configuration data containing header information (bitstream name, date, etc.) that is recognized by configuration tools but not written to the FPGA. Use the Fabric Configuration tool to directly write the .sbit file to the FPGA through the Cable
.bin	Binary configuration data without headers (pure bit stream) for user configuration schemes such as microprocessor to FPGA configuration.
.sfc	Bitstream file written to Flash, converted from .sbit file

The .sbit file is the default bitstream file generated by the PDS tool after compilation, as shown below, run Generate Bitstream to generate the .sbit file.

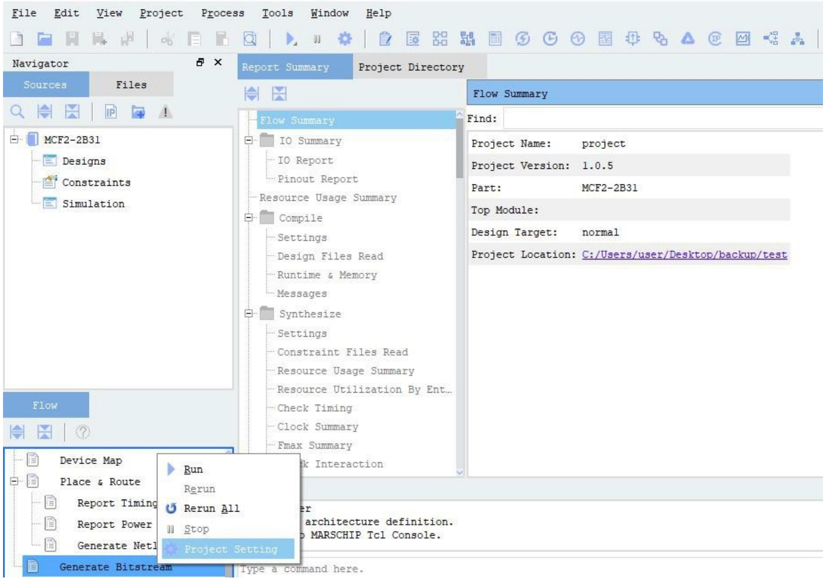


Figure 26 Schematic diagram of the generated bit stream

In the above picture, right click on Generate Bitstream, select Project Setting, the following picture will pop up, check Create Bin File, click ok and then re-run Generate Bitstream, then generate both .sbit and .bin files.

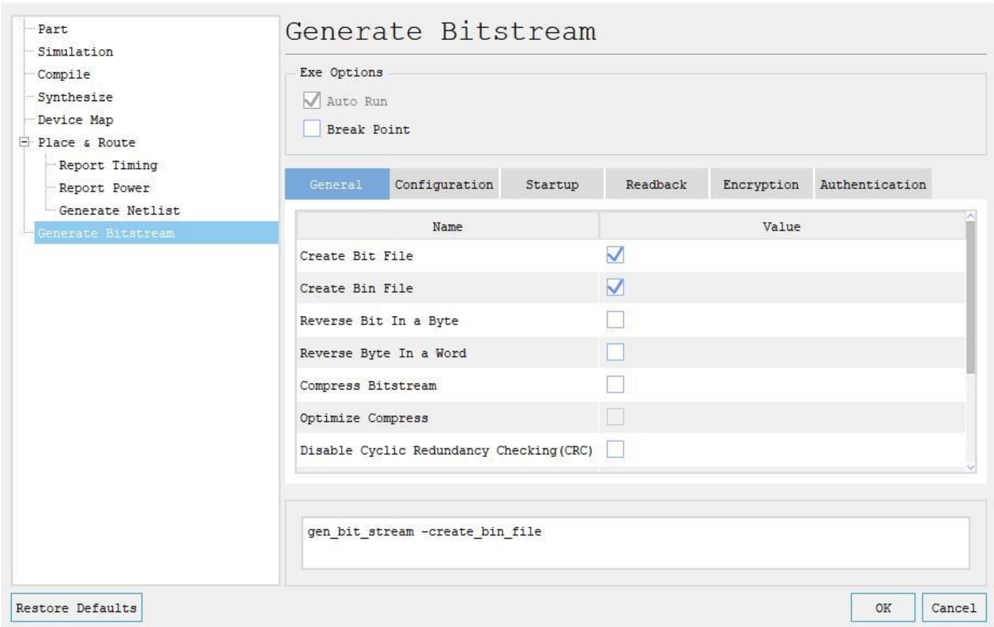


Figure 27 Configure Generate Bitstream

In the Fabric Configuration tool interface, click Operations in the menu bar, click Convert File in the drop-down menu, and the interface shown in the figure below will pop up.

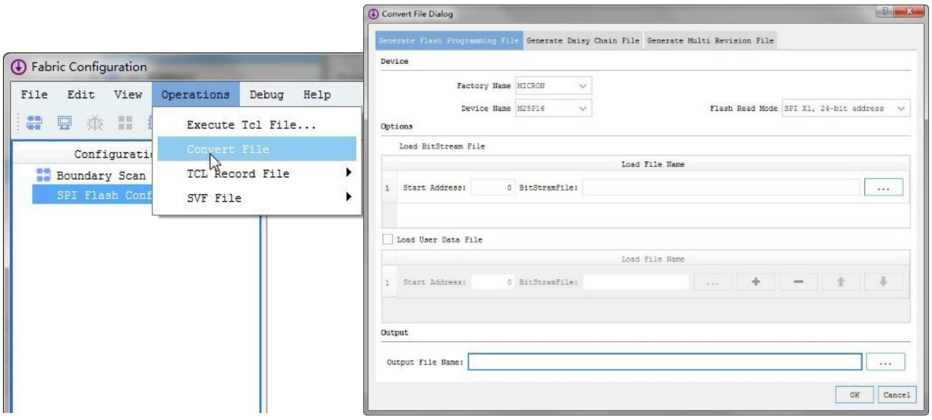


Figure 28 Generate Flash Programing File

Users select the corresponding Flash Device according to the actual application, select the Flash data bit width and address bit width in the Flash Read Mode drop-down box, then fill in the sbit file path and sfc file storage path, and click ok to generate the Flash configuration file-sfc file.

2. Bitstream size

The bitstream sizes for each device in the Titan2 family are shown in the table below. The compressed bitstream file is smaller than before the compression. The size of the compressed file is design dependent and is not described here.

Table 13 Bitstream sizes for each Titan2 family device

COMPONENTS	FILE NAME	UNCOMPRESSED FILE SIZE (MBYTE)
PG2T390H	*.sbit	12.30

3. Bit order at different interface bit widths

Titan2 series FPGA configuration supports multiple bit widths, different configuration modes of the same bit width, and its bit sequence, that is, the corresponding relationship between the high and low bits of the data bus and the bit stream sequence is consistent. For example, the bit sequence of Master SPI (x8) and Slave Parallel (x8) modes is the same. Taking the synchronization word (32'h01332D94) in the bit stream as an example, the high and low bits of the data bus and the sequence of the bit stream are shown in the table below.

Table 14 Synchronization word sending order at different bit widths

CLOCK CYCLE	HEX	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
D[1:0](x2)	2'b	00	00	00	01	00	11	00	11	00	10	11	01	10	01	01	00
D[3:0](x4)	4'h	0	1	3	3	2	D	9	4								
D[7:0](x8)	8'h	01	33	2D	94												
D[15:0](x16)	16'h	0133	2D94														
D[31:0](x32)	32'h	01332D94															

Note: D[x:0] in the table only indicates the high and low bits of the data bus, and the corresponding pins should be referred to the port list under different interface modes.

(III) Configuration packets

1. Package Format

There are two types of packages for configuring bitstreams for Titan2 series FPGAs: Type 1 and Type 2.

Type 1 packets are used for register reads and writes. Type 1 packets consist of a packet header section and a data section.

The type 1 packet header is a 32-bit word. The packet header is followed by the data. The data is in 32-bit words.

If the number of 32-bit words followed by the header specified in the packet header section is 0, then the packet has only the header section without the data section and is empty.

The type 1 package header format is shown in the table below.

Table 15 Type 1 Package header format

[31:29]	[28:27]	[26:22]	[21:0]
Header Type	Operation Code	Register Address	Number of 32-bit words
101	00:no operation; 01:write; 10:read; 11:hold	Address of the register	Number of 32-bit words followed by the package header

Type 2 packets are used to read and write configuration memory. The type 2 packet consists of a packet header section and a data section.

The type 2 packet header is a 32-bit word. The packet header is followed by the data. The data is in 32-bit words.

The type 2 packet must follow the type 1 empty packet. The type 2 packet does not specify a register address and uses the address specified by the preceding type 1 null packet. The type 2 packet header format is shown in the table below.

Table 16 Type 2 Package header format

[31:29]	[28:27]	[26:0]
Header Type	Operation Code	Number of 32-bit words
010	00:no operation; 01:write; 10:read; 11:hold	Number of 32-bit words followed by the package header

2. Configuration Register

The configuration registers of Titan2 series FPGAs are listed in the following table.

Table 17 Configuration Register Descriptions

NAME	R/W	ADDRESS	DESCRIPTION
CRCR	R/W	00000	CRC Register
IDR	R/W	00001	Device Identification Register
CMDR	R/W	00010	Command Register

CTRL0R	R/W	00011	Control register 0
CTRL1R	R/W	00100	Control register 1
CMEMIR	W	00101	Frame data input register
MFWRITER	W	00110	Multi-frame write register
CMEMOR	R	00111	Frame data output register
STATUSR	R	01001	Status Register
CHAINR	W	01010	Cascade Register
ADRR	R/W	01011	Frame address register
SBPIR	R/W	01100	SBPI Register
SEUR	R/W	01101	SEU control register
SEUSTATUSR	R	01110	SEU Status Register
IRSTCTRLR	R/W	01111	Hot start control register
IRSTADRR	R/W	10000	Hot start address register
WATCHDOGR	R/W	10001	Watchdog register
HSTATUSR	R	10010	History Status Register
CMASKR	R/W	10111	Control Mask Register
OPTION0R	R/W	11001	Option register 0
OPTION1R	R/W	11010	Option register 1
SEUADDR	R	11101	SEU Frame Address Register
SEUNADDR	R	11111	SEU lower frame address register

The detailed description of each configuration register is as follows.

1) CRC Register (CRCR)

Each write to the CRC register corresponds to a CRC check of the bitstream. If the value written to the CRC register is the same as the current CRC value calculated by CCS, then the CRC check passes. Otherwise, init_complete is pulled low.

2) Device Identification Register (IDR)

The device identifier. The IDCODE is written to the ID register before loading the bitstream, and the written IDCODE matches the device IDCODE before subsequent operations can continue. Each write to the ID register corresponds to a device ID check. If the value written to the lower 28 bits of the ID register is the same as the value of the lower 28 bits of the device ID, then the ID check passes. Otherwise, init_complete is pulled low.

Table 18 Titan2 Family FPGA Device Models

DEVICE MODEL	DEVICE IDENTIFICATION
Titan2 390H	X0702899

3) Command Register (CMDR)

The command register determines the next operation performed by the configuration control system.

Write at most one command to the command register per operation. You cannot have a packet header followed by multiple commands.

Table 19 CCS commands and their descriptions

COMMAND	CODE	DESCRIPTION
NOP	00000	No operation
RSTCRC	00001	Reset CRC - Reset CRC register
SWITCH	00010	Switching master mode to configure clock frequency Start the update master mode configuration clock frequency operation, the value is determined by oscfsel in CTRL0R
WCMEM	00100	Write configuration data Used before writing configuration data to the configuration memory via CMEMIR
MFWRITE	00101	Multi-frame writing Writes the currently written frame of data to the next consecutive frames. The number of frames is determined by the MFWRITER
RCMEM	00110	Read configuration data Used before reading back configuration memory configuration data via CMEMOR
SWAKEUP	00111	Start wake-up operation
SWAKEDOWN	01000	Turn off the wakeup operation
GUP	01001	Internal logic enable
GDOWN	01010	Internal logic disabled
DESYNC	01011	Go to Synchronization Used at the end of the configuration
RWD	01100	Restart Watchdog
RRBCRC	01101	Reset Readback CRC
RBCRC	01110	Readback CRC Check
IRST	01111	Hot Start Hot start invalid when version rollback

WCMEMDIS	10000	Write Configuration Memory Disable Used to disable configuration memory writes after the configuration data has been written
RCMEMDIS	10001	Read configuration memory disable Used to disable configuration memory reads after the configuration data has been read

4) Control register 0 (CTRL0R)

Table 20 Control Register 0 Description

POSITION	NAME	INITIAL VALUE	DESCRIPTION
[31:23]	Reserved		
[22]	otsd_en	1'b0	Over Temperature Shutdown Enable 0: Disabled 1: Enabled
[21:5]	Reserved		
[4]	fallback_en	1'b1	Version rollback enable
[3]	Reserved		
[2]	persist	1'b0	The configuration interface (Master SPI interface, Slave Parallel interface, SlaveSerial interface) reserved selection in user mode 1’b0: released for user use 1’b1: Reserved to continue to be used to configure the interface In user mode, when the internal slave parallel interface is enabled, the Slave Parallel interface and the SlaveSerial interface cannot be reserved and continue to be used for configuration interfaces
[1]	Reserved		
[0]	dec_en	1'b0	Decryption Enable

5) Control register 1 (CTRL1R)

Table 21 Control Register 1 Description

POSITION	NAME	INITIAL VALUE	DESCRIPTION
[31:5]	Reserved		
[4]	mask_en	1'b0	Variable memory location readback mask enable 0: disable shielding 1: enable masking
[3]	Reserved	1'b0	
[2]	osc_off_en	1'b0	Controls whether user logic shutdown is allowed OSC 0: Not allowed 1: Allowed

[1:0]	wrctrl	2'b00	External port security level. Controls shutdown reassignment, partial reassignment, local dynamic reassignment, and readback of configuration memory. After configured to disable, cannot be changed to enable unless reset 00: Reassignment enable, readback enable (default) 01: Reassignment enable, readback disable 1x: Reassignment Prohibited, Readback Prohibited

6) Control Mask Register (CMASKR)

The control mask register is used to mask the corresponding bits in control register 0 and control register 1. The default value is 32'd0. 0 is the mask. For example, if the value of the control mask register is 32'h0000_0003, the low 2 bits of control register 0 and control register 1 can write data, but the other bits cannot write data.

7) Option register 0 (OPTION0R)

Table 22 Option Register 0 Description

POSITION	NAME	INITIAL VALUE	DESCRIPTION
[31:24]	Reserved		
[23]	done_syn	1'b0	External done sync enable. Indicates whether the external input done is a sync signal for the output done. If it is a synchronous signal, the CCS wake-up module uses it directly. If it is an asynchronous signal, it needs to be synchronized before use and is usually used to delay the done function. 0:Not synchronized 1: Synchronization
[22:21]	t_sel	2'd0	Wake-up cycle length selection, i.e. the time interval between T1, T2, T3 2'b00: 1 2'b01: 2 2'b10: 4 2'b11: 1
[20:19]	done_sel	2'd0	done Signal pull-up moment selection 2'd0: T3 2'd1: T1 2'd2: T2 2'd3: T4
[18:17]	gwen_sel	2'd0	gwen signal pull-up moment selection 2'd0: T2 2'd1: T1 2'd2: T3 2'd3: T4
[16:15]	grsn_sel	2'd0	grsn signal pull-up moment selection 2'd0: T2 2'd1: T1 2'd2: T3 2'd3: T4

[14:13]	gouten_sel	2'd0	gouten signal pull-up moment selection 2'd0: T1 2'd1: T2 2'd2: T3 2'd3: T4
[12]	Reserved	1'b0	
[11]	wait_pll	1'b0	Wait for PLL lock enable on wake-up 0: Do not wait for PLL lock 1: Need to wait for the PLL to lock
[10]	Reserved	1'b0	
[9]	startup_uclk_en	1'b0	User clock wake-up enable 0: Wake-up using the clock used at configuration 1: Wake-up using user clock
[8:7]	Reserved	3'd0	
[6:0]	oscfsel	7'd0	Master Mode Configuration Clock CFG_CLK Frequency Selection

8) Option Register 1 (OPTION1R)

Table 23 Select Register 1 Description

POSITION	NAME	INITIAL VALUE	DESCRIPTION
[31:2]	Reserved		
[1]	rbcrc_disable	1'b0	Disable readback CRC error indication 0: Enabled 1: Disabled
[0]	crc_disable	1'b0	Disable CRC check 0: Enabled 1: Disabled

9) Frame data input register (CMEMIR)

Configuration memory configuration data interface. Data is written to the configuration memory via CMEMIR.

10) Multi-Frame Write Register (MFWRITER)

The number of compressed frames indicates that the data content of the following consecutive frames is the same as the data content of the one frame just written. For example, after configuring one frame of data, if the data content of the next three consecutive frames is the same as this one, then the content of the multi-frame write register needs to be written as 3.

11) Frame data output register (CMEMOR)

Configuration Memory Readback Data Interface. Data is read back from the configuration memory via CMEMOR.

12) Status Register (STATUSR)

Table 24 Status Register Descriptions

POSITION	NAME	DESCRIPTION
[31:30]	Reserved	
[29]	prcfg_over	Local reassignment completion mark
[28]	prcfg_err	Partial reassignment error flag
[27]	over_temp	Over temperature sign
[26]	flg_x32	Slave Parallel mode 32-bit data bit width indication
[25]	flg_x16	Slave Parallel mode 16-bit data bit width indication
[24]	flg_x8	Slave Parallel mode 8-bit data bit width indication
[23:22]	ipal_m[1:0]	Internal data bit width selection from parallel interface
[21]	fallback	Backward indication sign
[20]	Reserved	
[19]	pll_lock	PLL Lock Flag
[18]	gwen	Global Write Enable
[17]	grsn	Global register set reset
[16]	gouten	Global IO output enable
[15]	glogen_fb	Global logic enable feedback
[14]	glogen	Global logic enable
[13]	done_i	DONE pin input
[12]	done	Device wake-up success flag
[11]	init_n	INIT_FLAG_N pin input
[10]	init_complete	Initialization completion and configuration error indication
[9:7]	m[2:0]	Mode Selection
[6]	wakedown_over	Wake-up shutdown end
[5]	wakeup_over	Wake-up call ends
[4]	timeout	Watchdog timeout
[3]	rbcrc_err	Read back CRC test results 0: CRC correct 1: CRC error

[2]	aut_err	Certification Results 0: Authentication passed 1: Authentication failed
[1]	crc_err	CRC test results 0: CRC correct 1: CRC error
[0]	id_err	ID test results 0: Correct 1: Incorrect

13) Watchdog register (WATCHDOGR)

Used for timeout detection in configuration mode and user mode.

Table 25 Watchdog Register Descriptions

POSITION	NAME	INITIAL VALUE	DESCRIPTION
[31]	Reserved	1'b0	
[30]	wd_cfg_en	1'b0	Watchdog enable in configuration mode
[29:0]	wd_value	30'h3FFF_FFFF	Watchdog timeout values

14) Frame Address Register (ADRR)

Provides the starting address for configuration and readback data. Only valid addresses can be sent, not out of range.

Table 26 Frame Address Register Description

POSITION	NAME	INITIAL VALUE	DESCRIPTION
[31:27]	Reserved		
[26:25]	type	2'd0	Configure memory content type 00: Non-DRM content 01: DRM Content 10: Reserved 11: Reservation
[24:20]	addr_region	5'd0	Regional Address
[19:18]	Reserved		
[17:10]	addr_column	8'd0	Column Address
[9:8]	Reserved		
[7:0]	addr_frame	8'd0	Frame address

15) SBPI Register (SBPIR)

Provides Master SPI mode control and option information.

Table 27 SBPI Register Descriptions

POSITION	NAME	INITIAL VALUE	DESCRIPTION
[31]	emclk_en	1'b0	External master clock enable 1'b0: Master mode configuration clock uses internal clock 1'b1: Master mode clock uses external clock
[30]	sbpi_rf_sel	1'b0	Data sampling clock edge selection 0: Rising edge 1: Falling edge Fast mode for falling edge sampling in Master SPI mode
[29:11].	Reserved		
[10]	addrwidth	1'b0	SPI Address Bit Width 0: 24-bit address 1: 32-bit address
[9:8].	datawidth	2'd0	SPI data bit width 00: X1 01: X2 10: X4 11: X8
[7:0]	opcode	8'h0B	SPI master mode opcode

16) SEU control register (SEUR)

Provides SEU error checking and error correction and readback CRC control.

Table 28 SEU Control Register Descriptions

POSITION	NAME	INITIAL VALUE	DESCRIPTION
[31:2]	Reserved		
[1]	seu_en	1'b0	SEU Enable
[0]	rbcrc_en	1'b0	Readback CRC Enable

17) SEU Status Register (SEUSTATUSR)

Table 29 SEU Status Register Descriptions

POSITION	NAME	DESCRIPTION
[31:17]	Reserved	
[16]	drcfg_over	Dynamic reallocation completion mark
[15]	drcfg_err	Dynamic reallocation error flag

[14:3]	seu_index	SEU Single-bit error position indication in the frame
[2]	seu_ded	SEU double bit error flag
[1]	seu_sec	SEU single-bit error flag
[0]	seu_over	SEU completion mark

18) SEU Frame Address Register (SEUADDR)

Table 30 SEU frame address register description

POSITION	NAME	DESCRIPTION
[31:21]	Reserved	
[20:16]	seu_region_naddr	SEU Area address of the current frame
[15:8]	seu_column_naddr	SEU Column address of the current frame
[7:0]	seu_frame_naddr	SEU Frame address of the current frame

19) SEU lower frame address register (SEUNADDR)

Table 31 SEU lower frame address register description

POSITION	NAME	DESCRIPTION
[31:21]	Reserved	
[20:16]	seu_region_naddr	SEU Area address of the next frame
[15:8]	seu_column_naddr	SEU Column address of the next frame
[7:0]	seu_frame_naddr	SEU Frame address of the next frame

20) Hot Start Control Register (IRSTCTRLR)

Hot start control.

Table 32 Hot Start Control Register Description

POSITION	NAME	INITIAL VALUE	DESCRIPTION
[31:1]	Reserved		
[0]	vback_en	1'b0	Version fallback enable. Controls whether the fallback is to the gold bitstream or the previous version of the bitstream 0: Disabled 1: Enabled

21) Hot Start Address Register (IRSTADRR)

FLASH first address at hot boot, byte addressing.

Table 33 Hot Start Address Register Description

POSITION	NAME	INITIAL VALUE	DESCRIPTION
[31:24]	FLASH First Address	8'd0	FLASH first address [31:24] bits for 4-Byte mode
[23:0]	FLASH First Address	24'd0	FLASH First Address

22) History Status Register (HSTATUSR)

Table 34 History Status Register Description

POSITION	NAME	DESCRIPTION
[31:15]	Reserved	
[14]	timeout1	Previous status timeout 0: Watchdog not timed out 1: Watchdog timeout
[13]	rbcrc_err1	Previous state readback CRC error 0: Correct 1: Error
[12]	crc_err1	Previous status CRC error 0: Correct 1: Error
[11]	id_err1	Previous status ID error 0: Correct 1: Error
[10]	irst1	The previous status version is a hot start upgrade version indication 0: Non-hot start upgrade version 1: Hot start upgrade version
[9]	fallback1	Indicates whether the previous state is a fallback operation 0: No 1: Yes
[8]	valid1	Previous status valid 0: Invalid 1: Valid
[7]	1'b0	Reserved
[6]	timeout0	Current status timeout 0: Watchdog has not timed out 1: Watchdog timeout

[5]	rbcrc_err0	Current state readback CRC error 0: Correct 1: Error
[4]	crc_err0	Current Status CRC Error 0: Correct 1: Error
[3]	id_err0	Current Status ID Error 0: Correct 1: Error
[2]	irst0	The current status version is a hot start upgrade version indication 0: Non-hot start upgrade version 1: Hot start upgrade version
[1]	fallback0	Indicates whether the current state is a fallback operation 0: No 1: Yes
[0]	valid0	Current status valid 0: Invalid 1: Effective

(IV) Bitstream format

This section lists common bitstream formats.

Table 35 Common Bitstream Format

FFFFFFFF	Filler words (100)
......	
FFFFFFFF	
000000AA	Automatic bus bit width detection
08100020	
FFFFFFFF	Filler words (10)
......	
FFFFFFFF	
01332D94	Synchronization
AB000001	Type 1 packet header: Write SBPIR register
xxxxxxxx	Data: SBPI Master Mode Configuration Control Options
A0000000	No operation type 1 package head (10 pcs)
......	
A0000000	
ABC00001	Type 1 packet header: write IRSTCTRLR register
xxxxxxxx	Data: Contents of IRSTCTRLR
AC000001	Type 1 packet header: write IRSTADRR register
xxxxxxxx	Data: Contents of IRSTADRR
A8800001	Type 1 packet header: Write CMDR register
00000000	Data: NOP command
A8800001	Type 1 packet header: Write CMDR register
00000001	Data: RSTCRC command
A8400001	Type 1 packet header: Write IDR register
xxxxxxxx	Data: IDCODE
AC400001	Type 1 packet header: Write WATCHDOG register
xxxxxxxx	Data: WATCHDOG
ADC00001	Type 1 packet header: Write CMASKR register
xxxxxxxx	Data: CMASKR's content
A8C00001	Type 1 packet header: Write CTRL0R register
xxxxxxxx	Data: Contents of CTRL0R
AE400001	Type 1 packet header: Write OPTION0R register
xxxxxxxx	Data: Content of OPTION0R
AE800001	Type 1 packet header: Write OPTION1R register
xxxxxxxx	Data: Contents of OPTION1R
A8800001	Type 1 packet header: Write CMDR register
00000002	Data: SWITCH commands
A0000000	No operation type 1 package head (10 pcs)
......	
A0000000	

AAC00001	Type 1 packet header: Write ADRR register
xxxxxxxx	Data: Contents of ADRR
A8800001	Type 1 packet header: Write CMDR register
00000004	Data: WCMEM commands
A9400000	Type 1 packet header: Write CMEMIR register
{3'b010, 2'b01, 27'dx}	Type 2 package header: write operation
xxxxxxxx	Data: Write the contents of the configuration memory
......	
xxxxxxxx	
A0000000	No operation type 1 package head (50 pcs)
......	
A0000000	
A8800001	Type 1 packet header: Write CMDR register
00000010	Data: WCMEMDIS command
A8800001	Type 1 packet header: Write CMDR register
00000012	Data: VDDTUP command
A8000001	Type 1 packet header: Write CRCR register
xxxxxxxx	Data: CRC value
ADC00001	Type 1 packet header: Write CMASKR register
xxxxxxxx	Data: CMASKR's content
A9000001	Type 1 packet header: Write CTRL1R register
xxxxxxxx	Data: Contents of CTRL1R
A8800001	Type 1 packet header: Write CMDR register
00000009	Data: GUP command
A8800001	Type 1 packet header: Write CMDR register
00000007	Data: SWAKEUP command
A8000001	Type 1 packet header: Write CRCR register
xxxxxxxx	Data: CRC value
A8800001	Type 1 packet header: Write CMDR register
0000000B	Data: DESYNC command
A0000000	No operation type 1 package head (100 pcs)
......	
A0000000	

(V) Download line rate

When using USB Cable for downloading, including JTAG downloading and SPI Flash programming, the default rate is 10MHz. The download rate can be set in Fabric Configuration, click the icon Connect To Server.

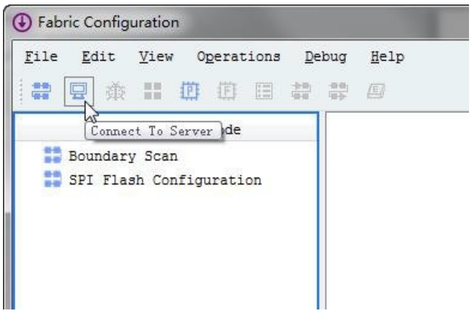


Figure 29 Fabric Configuration

If not connected, connect first, as shown on the left below, and if connected, click Next, as shown on the right below:

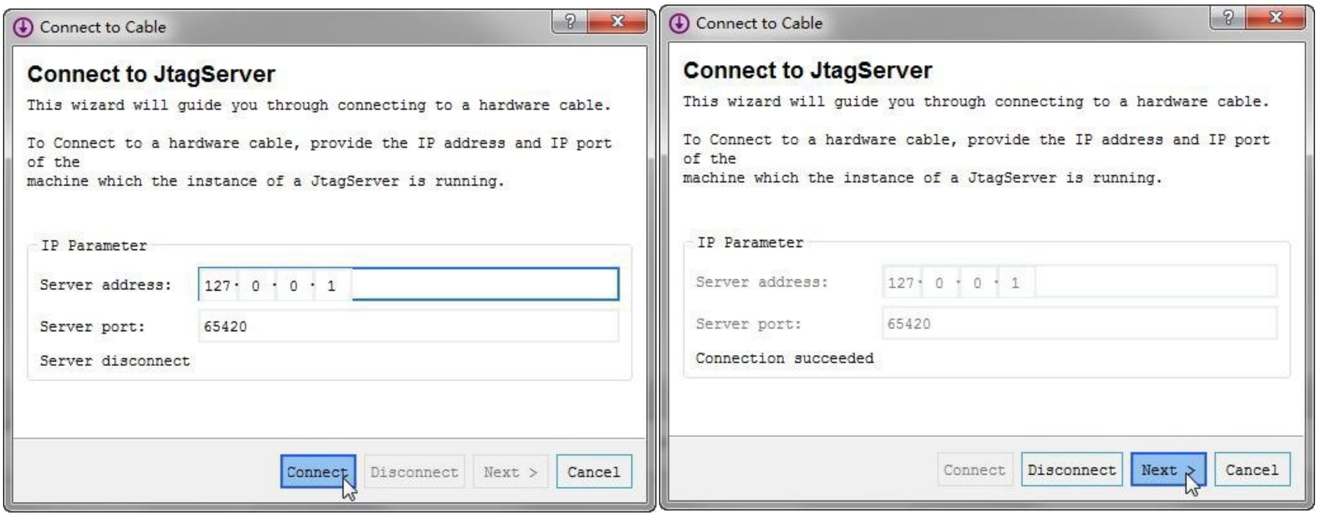


Figure 30 Connect To Cable

The USB Cable frequency can be selected in the TCK Frequency drop-down box.

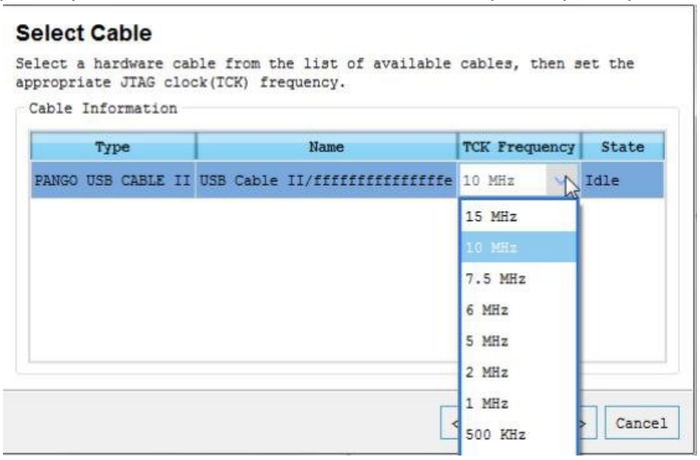


Figure 31 USB Cable Rate Settings

(VI) Configuration rate

1. JTAG mode

JTAG mode download at up to 50MHz.

Due to the rate limitation of the PANGO USB CABLE II download cable, the maximum configuration frequency is 15MHz when downloading using the download cable.

2. Master SPI Mode

Master SPI mode, CFG_CLK clock frequency is configurable, the default is 2.99MHz. low speed mode (i.e. rising edge sampling) supports a maximum configuration frequency of 15.38MHz, high speed mode (i.e. falling edge sampling) supports a maximum configuration frequency of 40MHz. master SPI serial daisy chain supports a maximum of 22.22MHz.

3. Slave Parallel Mode

In Slave Parallel mode, the download clock comes from the external Host device and the clock rate can be up to 50MHz.

4. Slave Serial Mode

In Slave Serial mode, the download clock comes from an external Host device and can be set to a maximum of 50MHz.

5. Configuration time

The configuration time required for different configuration modes can be roughly estimated based on the configuration clock frequency, bitstream size, and configuration bit width, as shown in the following equation:

$$\text{Configuration time (s)} \approx \frac{Bitstream\ size\ (bit)}{Clock\ frequency\ (Hz) \times Configuration\ bit\ width}$$

where the JTAG mode configuration bit width is 1.

(VII) Multi-start

When using the multiboot feature, the external Flash needs to support at least 2 sets of bitstreams. Bitstream 0 is the gold bitstream version or the application bitstream version with a start address of 0. The other bitstreams are the application bitstreams. Any error occurs during configuration, reset the version to fallback the circuit outside of the circuit and reload the gold bitstream or the previous version of the application bitstream. If there is also an error in the gold bitstream or the previous version of the application bitstream, then it is no longer reset, but FCS_N is set to 1 while INIT_FLAG_N is set to 0. The Master SPI mode operation ends.

When using the multiboot feature, the watchdog needs to be turned on. During configuration, the following errors will trigger a version rollback.

1. Device ID error

2. CRC error

3. Watchdog timeout

4. Authentication failure

1. Gold bit stream initialization system

In this application scenario, the FPGA will first boot from the gold bitstream and enter user mode, then complete the bitstream update of the Flash and the hotstart jump to the application bitstream operation via JTAG or user logic. When generated in software, the difference with the application bitstream initialization system below is both that the insert embedded hot boot is not checked. (Insert Embedded Hot Boot settings: Fabric Configuration->Operations->Convert File->Generate Multi Revision File->Data Stream Type:Multi Boot Data Stream ->Insert IRST CMD to Data Stream)

1. Load the gold bit stream from the Flash 0 address.

2. Use JTAG or user logic for Flash operation. If the application bitstream used for multiboot is already in Flash, Flash operation can be performed without it.

3. Hot start using JTAG/internal from parallel interface.

4. The FPGA loads the application bitstream from the Flash address specified in the IRSTCTRLR register.

5. If there is an error in the process of loading the application bitstream, load the golden bitstream from the Flash 0 address or fall back to the previous version of the application bitstream.

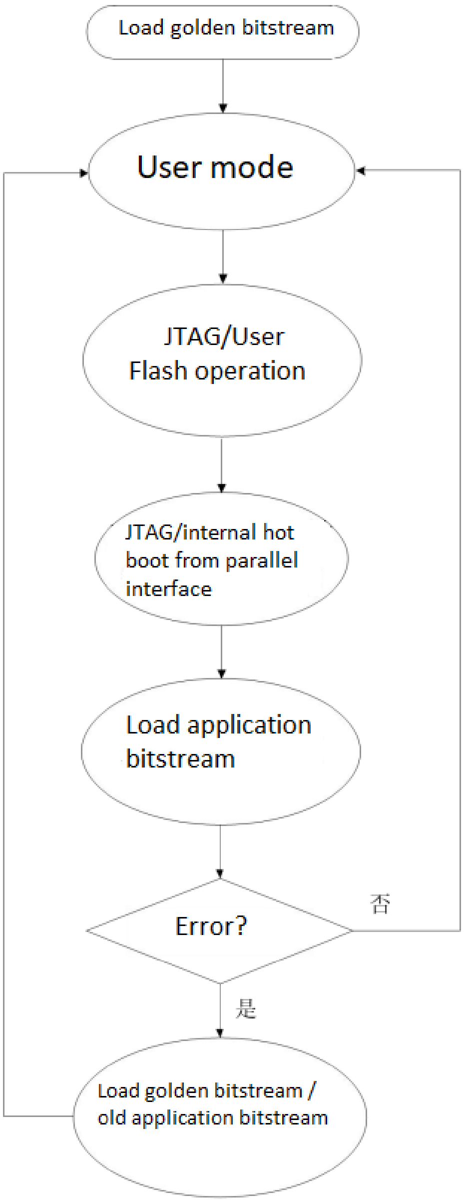


Figure 32 Gold bit stream loading flow

2. Apply bitstream initialization system

In this application scenario, the FPGA will first boot from the specified application bitstream and enter user mode, which is more convenient compared to the golden bitstream initialization system, which does not require additional operations to boot from the application bitstream. The difference with the golden bitstream initialization system is that there is a check box to insert embedded hot start when generating in software. (Insert embedded hot start settings: Fabric Configuration->Operations->Convert File->Generate Multi Revision File->Data Stream Type:Multi Boot Data Stream->Insert IRST CMD to Data Stream)

1. Load the gold bit stream from the Flash 0 address.

2. Gold bitstream embedded hot start enables loading the application bitstream from the Flash address specified by IRSTCTRLR.

3. If there is an error during loading the application bitstream, load the golden bitstream from the Flash 0 address or fall back to the previous version of the application bitstream.

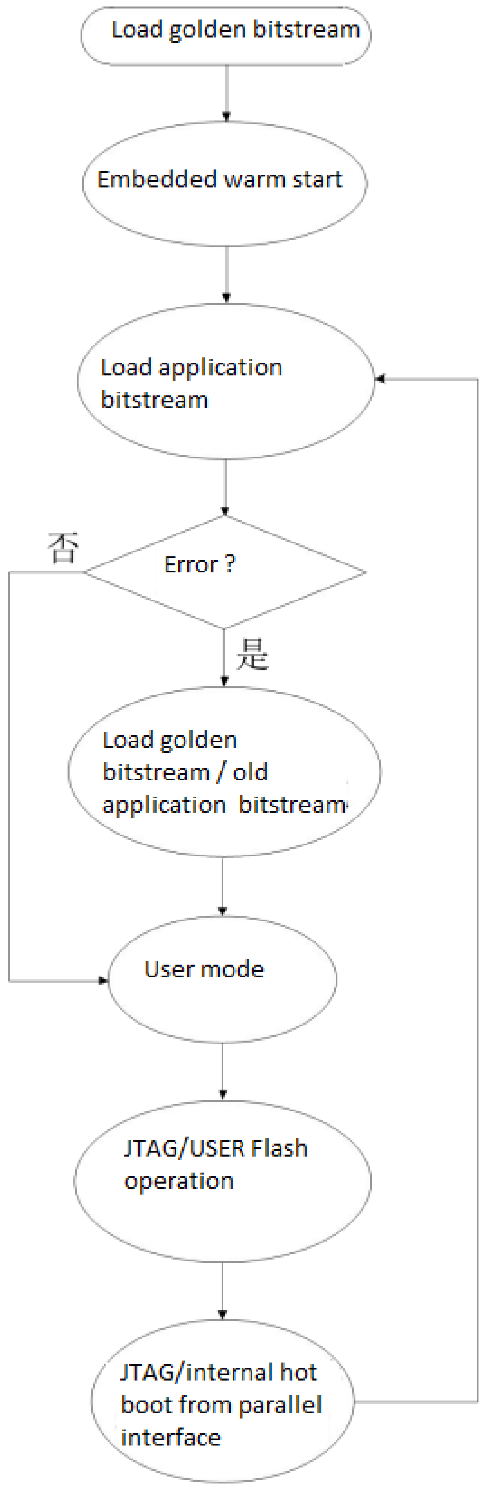


Figure 33 Application Bitstream Loading Flow

3. No golden bitstream

If Flash supports storing at most 2 sets of bitstreams, when version fallback is enabled, both sets of bitstreams are application bitstreams.

1. Load the application bitstream from the Flash 0 address.

2. JTAG or the user performs Flash operations. (Optional) If the application bitstream used for multiboot is already in Flash, Flash operation can be left out.

3. JTAG hot start or internal hot start from parallel interface.

4. Load the application bitstream from the Flash address specified in the IRSTCTRLR register.

5. If there is an error in the process of loading the application bitstream, fall back to the previous version of the application bitstream.

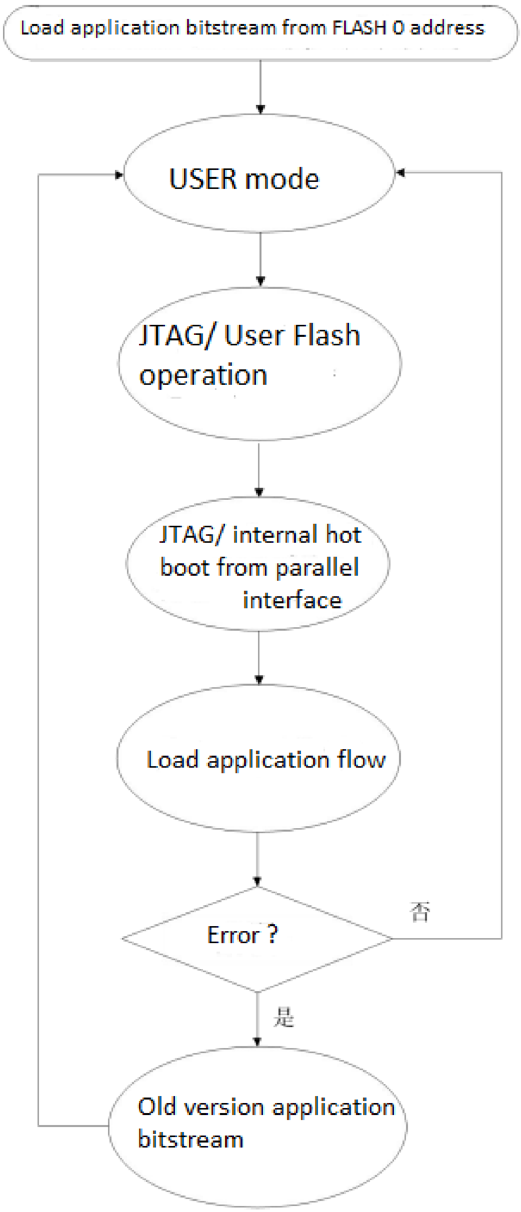


Figure 34 Loading flow when no gold bit flow

4. Hot Start

Before sending commands to the device, synchronization is required to ensure that the device can correctly resolve the command order. If the instruction is sent through the Slave Parallel interface, a bus bit width detection instruction is also required. In multiboot applications, the gold bitstream

contains a write IRSTADRR register instruction, and the write address is the application bitstream start address, so it is not necessary to include a write IRSTADRR register operation during hot boot. If the multiboot does not contain a gold bitstream, the user is required to write to the IRSTADRR register in the hotboot process to write the start address of the bitstream loaded during hotboot. If the gold bitstream has a hot start instruction embedded in it, i.e. in the application bitstream initialization system, it is not possible to upgrade to the gold bitstream via hot start.

Hot start can also be used to clear configuration memory, such as when lifting the JTAG temporary ban, when the IRSTADRR register does not need to be written. The complete hot start flow is shown in the table below.

Table 36 Hot Start Process

FFFFFFFF	Filler words (100)
......	
FFFFFFFF	
000000AA	Automatic bus bit width detection
08100020	
FFFFFFFF	Filler words (10)
......	
FFFFFFFF	
01332D94	Synchronization
AC000001	Type 1 packet header: write IRSTADRR register (optional)
xxxxxxxx	Data: Contents of IRSTADRR (optional)
A8800001	Type 1 packet header: Write CMDR register
0000000F	Data: IRST command
A8800001	Type 1 packet header: Write CMDR register
0000000B	Data: DESYNC command
A0000000	No operation type 1 package head (100 pcs)
......	
A0000000	

5. Watchdog

The watchdog is used to detect the timeout of the configuration process.

In configuration mode, the watchdog clock is the configuration clock. Each clock cycle, the watchdog count is decremented by 1. When the watchdog count is decremented to 0, the watchdog timeout flag is generated.

If the version fallback is enabled in the bitstream settings (Project Setting->Generate Bitstream->Configuration->Enable Watchdog In Configuration Mode) and the appropriate watchdog value is set (Load Watchdog), the watchdog timeout will trigger a version fallback during the Master SPI configuration.

The watchdog is disabled during and after the version rollback is completed. The hotstart can unblock the watchdog after the version rollback is successfully completed.

Watchdog Values

The watchdog is valid in different configuration modes, and when using different configuration

modes, the watchdog also uses the corresponding clock, so an estimate of the watchdog value can be obtained from the actual bitstream size ($L_{bitstream}$) as well as the configuration bit width (W).

Namely: Watchdog (Hex) > $L_{bitstream}/W$

IV. Readback operation

The Titan2 FPGA supports readback via JTAG and Slave Parallel interfaces.

(I) JTAG interface readback

The JTAG interface readback can be implemented through the Fabric Configuration tool.

(II) Slave Parallel Interface Readback

To use the Slave Parallel interface for readback, make sure the interface is enabled or reserved.

If you are using this interface in configuration mode for readback, you should ensure that the configuration mode currently in effect is Slave Parallel mode, i.e. MODE[2:0] = 3'b110.

If you are using this interface for readback in user mode, you should make sure that the currently active bitstream has a reserved configuration interface set (set at Project Setting->Generate Bitstream->Readback->Persist Pin in PDS, not reserved by default) and that it is configured via the Slave Parallel interface. And the data bit width used for readback must be the same as the bit width used for configuration, i.e. if the configuration is done with Slave Parallel x8 and the configuration interface is reserved, then the readback can be done through Slave Parallel x8, but not with x16/x32 bit width.

The Slave Parallel interface needs to detect the wakeup end signal wakeup_over in the status register before readback operation to ensure wakeup before readback. Usually the readback is done long after the wakeup has ended.

When using the Slave Parallel interface for readback, it is necessary to first send some instructions to the device according to the Slave Parallel interface timing requirements, see readback configuration memory flow and readback configuration register flow, where the instruction stream is presented in the form of 32bits and needs to be sent according to the bit order under different interface bit widths. After receiving the instructions correctly, the device will send out the data on the rising edge of the clock when the data is ready, and the user can sample the data. The read-back configuration register and configuration memory processes are different, as described below. Refer to the typical timing diagram of the Slave Parallel configuration interface in Chapter 2 for the timing requirements to be met for read-back.

1. Read back configuration memory

Table 37 Readback configuration memory flow

PROCESS	COMMAND FLOW (HEX)/OPERATION
100 filler words	FFFFFFFF ...... FFFFFFFF
Automatic bus bit width detection	000000AA 08100020

10 filler words	FFFFFFFF ...... FFFFFFFF
Synchronized Word	01332D94
Type 1 package header: empty operation	A0000000
Write ADRR register	AAC00001 00000000(1)
Write RCMEM command to CMDR register	A8800001 00000006
Write type 1 packet header for reading CMEMOR registers	B1C00000
Write type 2 packet header for reading data	5xxxxxxx(2)
Suspend operation for at least 40 configured clock cycles (CS_N is 1, clock CFG_CLK is continuously on)	Pull CS_N high and change RWSEL to read when CS_N is 1(3)
Read CMEMOR register	Pull CS_N low, sample data, pull CS_N high after reading, change RWSEL to write(3)
Write RCMEMDIS command to CMDR register	A8800001 00000011
Write NOP command to CMDR register	A8800001 A0000000
Write DESYNC command to CMDR register	A8800001 0000000B
100 no operation type 1 package head	A0000000 ...... A0000000

Note: (1) Write the readback start address to the ADRR register.
(2) The number of 32bits of words to read back.
(3) The timing requirements to be met for RWSEL changes are described in Slave Parallel Readback Timing.

2. Read back configuration registers

The flow of reading back the configuration registers is shown below:

Table 38 Readback configuration register flow

PROCESS	COMMAND FLOW (HEX)/OPERATION
100 filler words	FFFFFFFF ...... FFFFFFFF
Automatic bus bit width detection	000000AA 08100020
10 filler words	FFFFFFFF ...... FFFFFFFF

Synchronization	01332D94
Type 1 package header: empty operation	A0000000
Write NOP command to CMDR register	A8800001 A0000000
Write type 1 packet header for reading the specified register	Bxxxxxxx(1)
Wait a minimum of 20 configuration clock cycles	Pull CS_N high and change RWSEL to read when CS_N is 1(2)
Read the specified register	Pull CS_N low, sample data, pull CS_N high after reading, change RWSEL for writing (2).
Write DESYNC command to CMDR register	A8800001 0000000B
100 no operation type 1 package head	A0000000 ...... A0000000

Note: (1) According to the packet format, fill in the address of the register to be read back and the number of 32bits words to be read back.

(2) The timing requirements to be met for RWSEL changes are described in Slave Parallel Readback Timing.

V. SEU testing

The Titan2 FPGA supports SEU 1-bit error correction and 2-bit error detection. the ECC algorithm uses the SECDED algorithm, grouped by frame. SEU error detection and error correction can be performed internally from the parallel interface.

In the SEU error detection and correction process, the user logic should save the current readback frame, and when a single bit error occurs, the user logic flips the error bit and then dynamically reassigns the frame for error correction.

When a double-bit error occurs, the location of the error cannot be detected and can only be corrected by dynamically reassigning the correct bitstream frame. Since the error may occur in any frame, if the user needs double-bit error correction, he needs to ensure that the correct bitstream is available, such as saved in SPI FLASH, and read the correct bitstream frame to dynamically reassign for error correction in case of double-bit error.

Titan2 SEU function can be implemented using IP, refer to the document "UG042001_SEU_IP" for details.

VI. Design protection

(I) Bitstream encryption

Malicious plagiarism of user designs by competitors is one of the common problems in the use of FPGAs. To solve this problem, Titan2 series FPGAs have a built-in bit stream encryption protection function to prevent users' designs from being maliciously cloned.

1. Scenarios without bitstream encryption protection

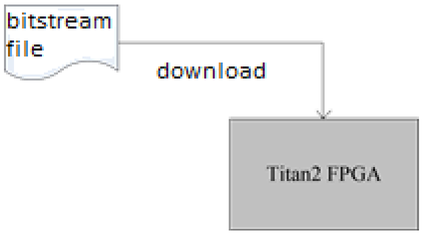


Figure 35 Bitless stream encryption scenario

As shown above, the Titan2 series FPGAs are SRAM-based FPGAs that require bitstream files to be loaded externally each time they are powered up. The user's bitstream file is usually stored in a memory cell external to the FPGA, such as Flash.

There are various ways for a malicious clone to obtain this bitstream file. After acquiring the bitstream file, the malicious clone loads the bitstream file directly onto the same FPGA they purchased themselves to achieve the exact same FPGA functionality as the user.

2. Scenarios with bitstream encryption protection

To effectively protect the user's intellectual property, the Titan2 family of FPGAs provides bitstream encryption.

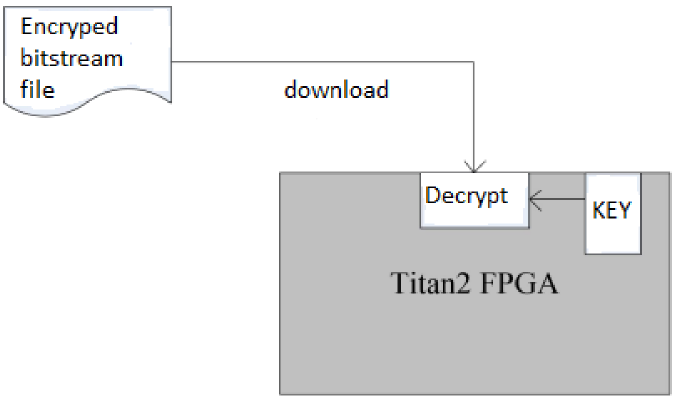


Figure 36 bit stream encryption scenario

As shown above, the bitstream file itself is encrypted, and when the encrypted bitstream file is loaded into the Titan2 Series FPGA, it must first be decrypted in conjunction with a key pre-stored inside the chip. Here, the key of the encrypted bitstream file and the key stored inside the Titan2 FPGA must match, otherwise the Titan2 FPGA will not be able to enter the working state properly.

The key inside the Titan2 FPGA is written in advance by the user via JTAG to the eFuse and power

supply RAM (BBRAM), or to a temporary key register for testing purposes. eFuse is one-time programmable (OTP) and non-volatile, so the key information is said to be retained permanently once programmed. In addition, the Titan2 FPGA key provides read protection. Keys written with read protection enabled cannot be read back, and the PDS download tool enables key read protection by default when the key is written to eFuse or BBRAM.

With the use of bitstream encryption protection, users are no longer bothered by malicious copying. Even if a malicious copier has the bitstream file (encrypted), he does not know what the key is, so he cannot load the bitstream file into a Titan2 FPGA that does not contain the corresponding key.

3. Encryption algorithms

Titan2 series FPGAs use AES256-GCM encryption algorithm to encrypt the bitstream, and the uncompressed bitstream also supports chunking encryption, which means the bitstream is divided into multiple chunks, each with a different key, and the key of each chunk is encrypted together with the previous chunk, except for the first chunk, which is normally stored in the FPGA. Chunked encryption provides stronger security against bypass attacks, such as DPA attacks. The number of chunked encryption chunks can be set through the PDS software. More chunks provide stronger security, but also increase the bitstream size, which affects the configuration time.

The AES256-GCM encryption algorithm also supports self-authentication. When self-authentication is turned on, the bitstream will not be loaded if the key is not known.

4. Flow of using bitstream encryption protection

The main processes used for bitstream encryption protection include:

1) Selecting the key, noting that the user needs to secure it properly;

2) User PDS generates an encrypted bitstream file using the selected key;

3) The user burns the key into the Titan2 family using the PDS built-in fabric configuration. Afterwards, the user can safely use the encrypted bitstream file to load the Titan2 FPGA after 3) processing.

To generate an encrypted bitstream file (sbit), right click on the Generate Bitstream option of the PDS and select the Project Setting item to set it, as shown below. You can also choose Process from the menu.

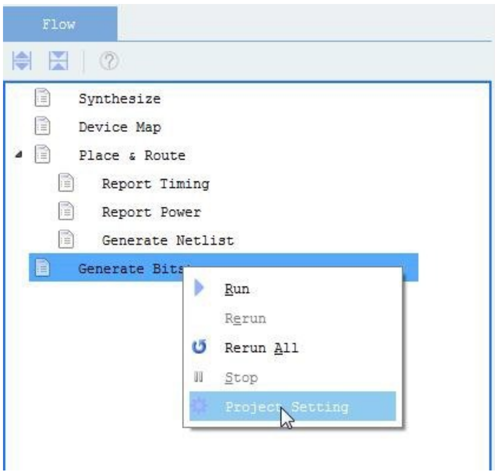


Figure 37 Configure interface

A pop-up window appears as follows:

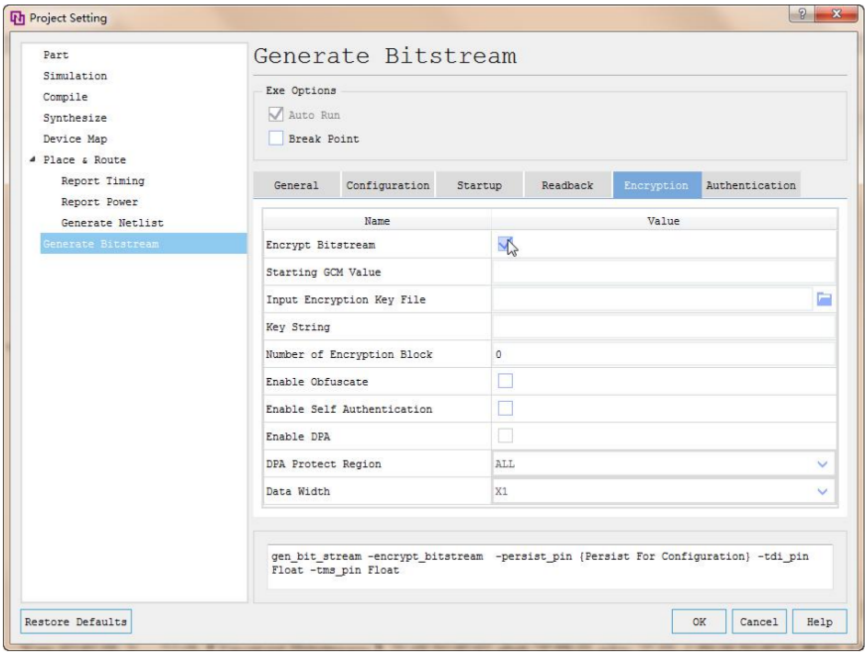


Figure 38 Configure Setup Options

Among them, **[Encrypt Bitstream]** selects whether to encrypt the bitstream file, and if **[Create Bit File]** of **[General]** is configured to Yes, check **[Encrypt Bitstream]** to generate the encrypted sbit file and nky file (including the initial GCM value of 96 bits required for encryption and 256 bit key). The user can edit the GCM and the key string or select the nky file, if the user does not specify it, the nky file will be generated randomly.

[Starting GCM Value] The user manually enters the initial GCM string;

[Input Encryption Key File] select nky file, if the key file is selected here, the software chooses to encrypt with the key file, regardless of whether the user has entered the key string in the options;

[Key String] The user manually enters the key string;

When the settings are complete, click OK to exit the configuration window. Then run Generate Bitstream to generate the encrypted bitstream file.

[Enable Obfuscate] Due to the openness of the AES algorithm standard, in order to enhance the security of the data flow, some optimization processing is performed on the basis of meeting the industry AES256 algorithm standard.

[Number of Encryption Block] The block size of the block encryption, in 128 frames (413696bits). The default value is 0, which means no block. The larger the value, the smaller the number of blocks.

[Enable Self Authentication] The self-authentication provided by the AES256-GCM encryption algorithm is enabled;

[Data Width] The encryption algorithm is in 128 bits. When configured with different configuration bit widths, the value here needs to be set according to the actual situation to ensure that the device can be decrypted properly.

(II) Readback protection

By default, Titan2 devices can be reconfigured and read back through the JTAG interface in user mode. If the internal slave parallel interface is instantiated in the user design, or if the internal slave parallel interface is not used and the Slave Parallel interface is reserved, it can Reconfiguration or readback is done internally from the parallel interface or the Slave Parallel interface. In order to prevent the configuration memory from being illegally accessed, the user can set not to reserve the external configuration interface, and not connect the internal slave parallel interface to the device pins. On the other hand, Titan2 devices also provide readback protection settings. After enabling readback protection, readback can only be enabled again by resetting the device.

See the figure below for the disabling of readback.

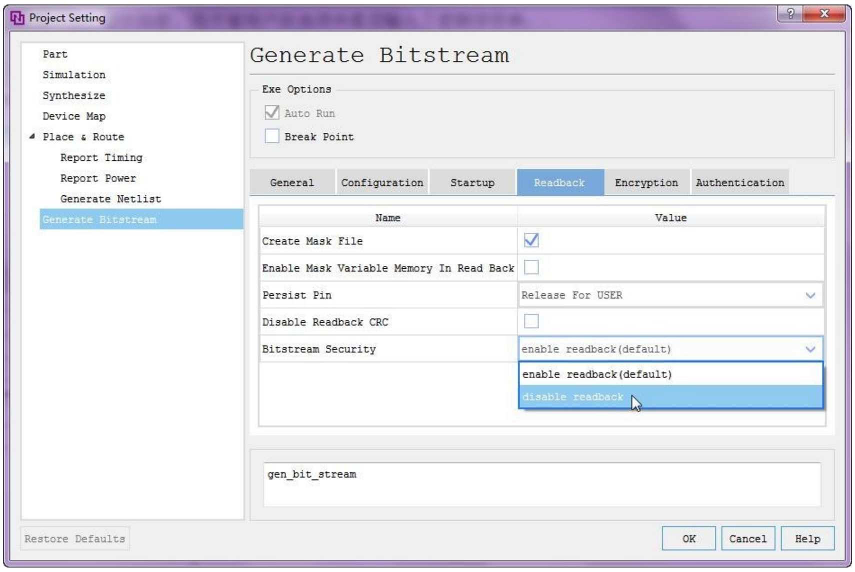


Figure 39 Readback disable setting

(III) Bitstream certification

In addition to the self-authentication provided by the AES256-GCM encryption algorithm, Titan2 also supports RSA authentication, using the authentication algorithm RSA-2048. The RSA-2048 public key digest algorithm is SHA3-384. The RSA-2048 public key digest length is 384 bits.

The bitstream digest algorithm is SHAKE256, and the bitstream digest length is 2047 bits. Authentication bitstreams support normal bitstreams, encrypted bitstreams and chunked encrypted bitstreams.

Authentication starts after the authentication register is written and ends after de-syncing.

While the authentication bit stream is authenticated, the net load is stored in the configuration memory. After the authentication is passed, subsequent wake-up operations can be performed. When the authentication fails, the authentication failure flag is generated and INIT_FLAG_N is set to 0.

The authentication encryption bit stream and the authentication chunk encryption bit stream are decrypted while authentication is performed, and the net load is stored in the configuration memory. After the authentication is passed, subsequent wake-up operations can be performed. When authentication fails, the authentication failure flag is generated and INIT_FLAG_N is set to 0.

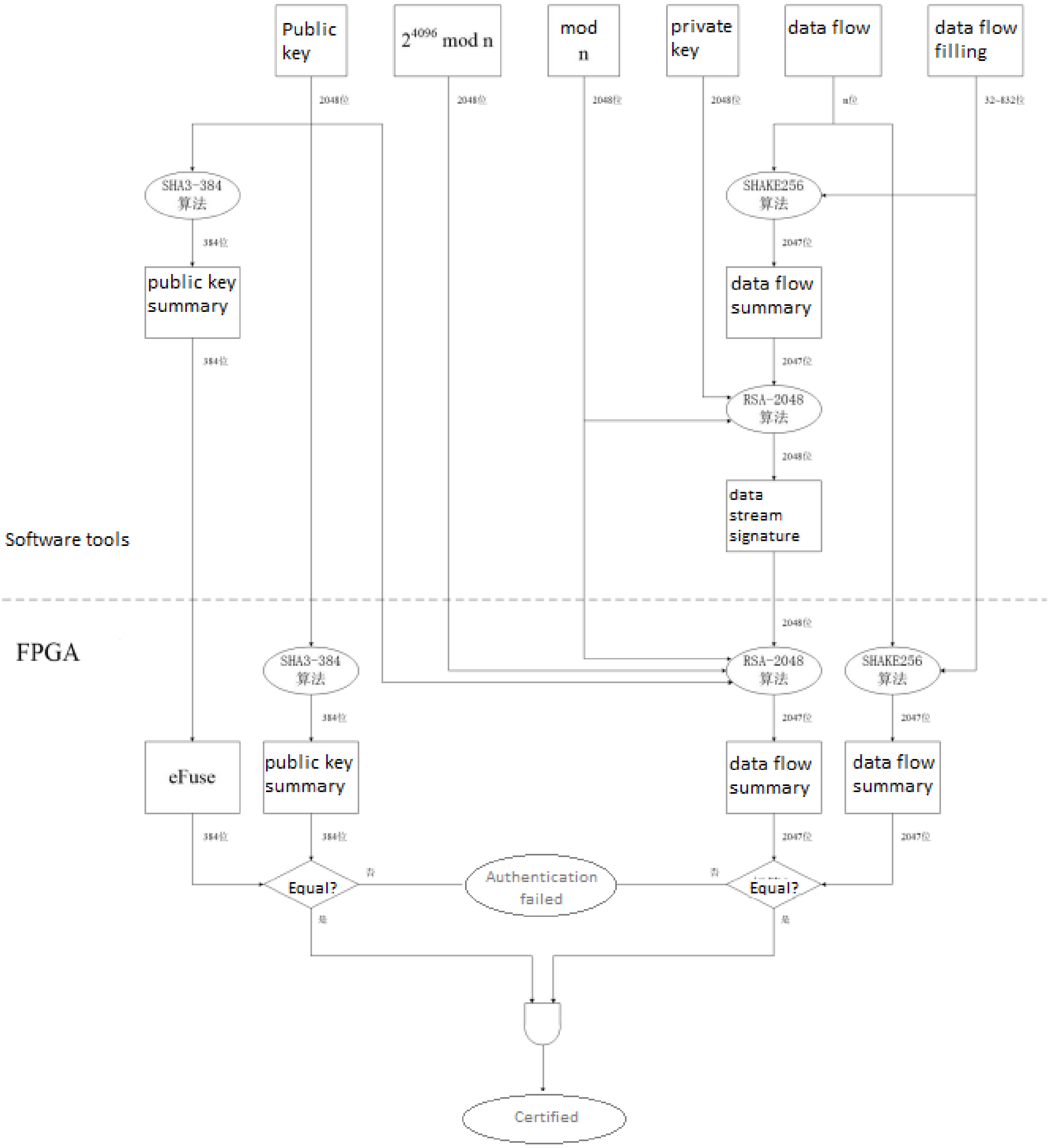


Figure 40 Schematic of the authentication process

(IV) DPA protection

Titan2 supports authentication methods such as AES256-GCM self-authentication, and RSA authentication. To further prevent DPA attacks, an effective way is to write the bitstream to the configuration memory without decrypting it during authentication, and decrypt and configure the bitstream only after the authentication is passed.

Titan2 provides DPA protection authentication support. When DPA protection is enabled, the DPA protection authenticates the encrypted bit stream for authentication while writing the encrypted net load to the configuration memory. After authentication, the encrypted net load is read back frame by

frame, decrypted and written back to the configuration memory, and then wake-up operation is performed. When authentication fails, the authentication failure flag is generated and INIT_FLAG_N is set to 0.

(V) UID

A major interest and safety problem in the reconfigurable device industry is design copy and unauthorized production. So in order to avoid such illegal behavior, UID came into being. Each device has a corresponding unique number, which has been uniquely determined when the device leaves the factory. Users can read through the UID interface (instantiated GTP_UDID) and JTAG interface, and after processing with their own unique encryption algorithm, the result will be incorporated into the programming bit stream. After reloading the bit stream every time, the device enters the user mode, and the user logic will first read the UID and process it with the user's unique encryption algorithm, then compare it with the result in the previously programmed bit stream, if there is any difference.The device will not work properly.

The 96-bit UID is stored in the eFuse and is programmed uniformly when the device is shipped. Each time the device is powered up, the UID in eFuse is automatically read into a register so that the user can retrieve it at any time.

(VI) JTAG security management

The JTAG interface provides users with a convenient and comprehensive method of configuring, reading back, and testing the chip. The JTAG interface can be disabled when the user wishes to keep the chip in a stable operating state free from malicious interference.

Titan2 provides support for secure management of the JTAG interface. JTAG can be temporarily disabled by instantiating the user JTAG interface in the user logic (corresponding to GTP_JTAGIF), which can be reopened by re-powering the chip or otherwise configuring the interface. Alternatively, JTAG can be permanently disabled by writing the JTAG disable bit in eFuse.

After the JTAG is temporarily disabled, it can be reopened through another interface. If GTP_IPAL_E2 is instantiated in the user logic, the user can send a hot start command through the internal Slave Parallel interface to reset the chip's configuration memory and reopen the external JTAG interface. If GTP_IPAL_E2 is not instantiated in the user logic and the external Slave Serial and Slave Parallel interfaces are retained, the external JTAG can be turned on by sending a hot start command through the external interface.

(VII) eFuse

eFuse is a one-time programmable non-volatile memory used in the Titan2 family of devices to store some device setup information, UIDs, and other user-programmable bits:

1. 256-bit AES-GCM encryption key

2. 384-bit public key digest

3. 32-bit user fuse

Security settings (authentication enable, DPA protection enable, key lock flag, public key digest lock flag, user Fuse lock flag). All user programmable bits are configurable via the PDS, as follows

The operation is described in the document Fabric Configuration User Manual.

Since the JTAG interface can be used to program the eFuse, it is recommended to lock all function bits of the eFuse after confirming the application scenario of the chip in order to avoid possible mis-writing of the eFuse due to environmental interference signals on the JTAG interface. The TCL command "cfg_efuse_lock -lock FF -password 000000" is used in the Fabric Configuration tool to complete the locking of all function bits.

The 96-bit UID is programmed uniformly when the device is shipped and can be read out by the JTAG interface or the UID interface, see (V) UID. 32-bit user

The fuse can be read via the JTAG interface or the user Fuse interface, see Chapter 9 (d) User Fuse Interface.

VII. User logic interface

(I) Internal slave parallel interface

GTP_IPAL_E2 provides an internal slave-parallel interface, and a read-back CRC and SEU interface. You can use this GTP to perform internal slave parallel interface operation, and you can also use the readback CRC and SEU interface to easily perform readback CRC and SEU operation. The SEU function can be implemented using IP, please refer to the document "UG042001_SEU_IP" for details.

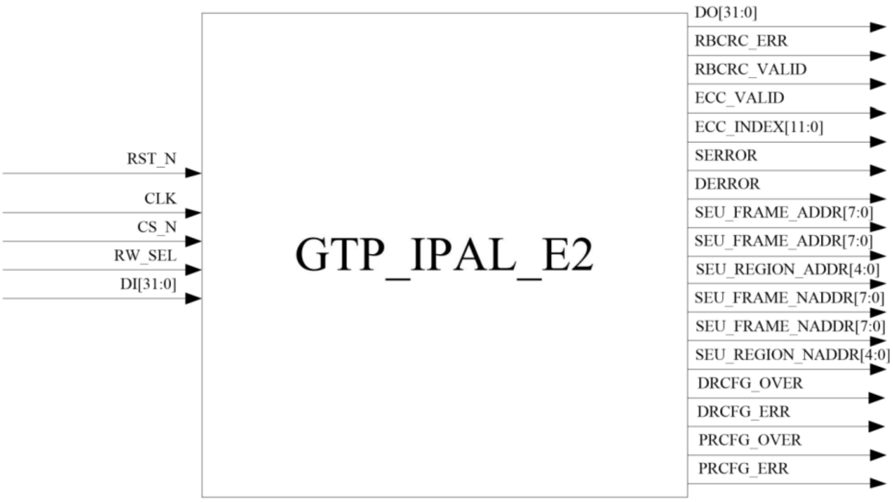


Figure 41 GTP_IPAL_E2 port diagram

1. Port List

Table 39 GTP_IPAL_E2 Port List

PORT NAME	I/O	BIT WIDTH	DESCRIPTION
CLK	I	1	Internal clock from parallel interface
RST_N	I	1	Reset signal, active low, for emulation only(1)
INTERNAL SLAVE PARALLEL INTERFACE			
CS_N	I	1	Chip select signal, active low
RW_SEL	I	1	Read/write options: 0 write, 1 read;
DI	I	32	Data input
DO	O	32	Data output
READBACK CRC INTERFACE			
RBCRC_ERR	O	1	Read back CRC error flag, active high.
RBCRC_VALID	O	1	Readback CRC valid flag, high valid for 1 clock cycle.
SEU DETECTION INTERFACE			

ECC_VALID	O	1	ECC valid mark, active high.
ECC_INDEX	O	12	Address index for single-bit errors
SERROR	O	1	Single-bit error flag detected by SEU
DERROR	O	1	Two-bit error flag detected by SEU
SEU_FRAME_ADDR	O	8	Current frame address detected by SEU
SEU_COLUMN_ADDR	O	8	Current address of the front of SEU detection
SEU_REGION_ADDR	O	5	Current zone address for SEU detection
SEU_FRAME_NADDR	O	8	Address of the next frame detected by SEU
SEU_COLUMN_NADDR	O	8	SEU detects the next column address
SEU_REGION_NADDR	O	5	Address of the next zone detected by SEU
PRCFG_OVER	O	1	The end of partial rewiring flag, which is 1, indicates the completion of partial rewiring.
PRCFG_ERR	O	1	Local reassignment error flag indicating the result of a local reassignment when PRCFG_OVER is active.
DRCFG_OVER	O	1	The dynamic reassignment end flag, when 1 indicates the completion of dynamic reassignment.
DRCFG_ERR	O	1	Dynamic reassignment error flag indicating dynamic reassignment result when DRCFG_OVER is active.

Note: (1) This signal is only used to restore the initial state of the interface during simulation, there is no such port on the actual hardware, and the operation of this port is not necessary and will not play any role.

2. Parameter Definition

Table 40 List of GTP_IPAL_E2 parameters

NAME	TYPE	VALID VALUES	DEFAULT VALUE	FUNCTION DESCRIPTION
IDCODE	Binary	0~32'hffffffff	32'haaaa5555	Device IDCODE, for simulation only
DATA_WIDTH	String	"X8", "X16", "X32"	"X8"	Parallel data bit width

3. Interface Timing

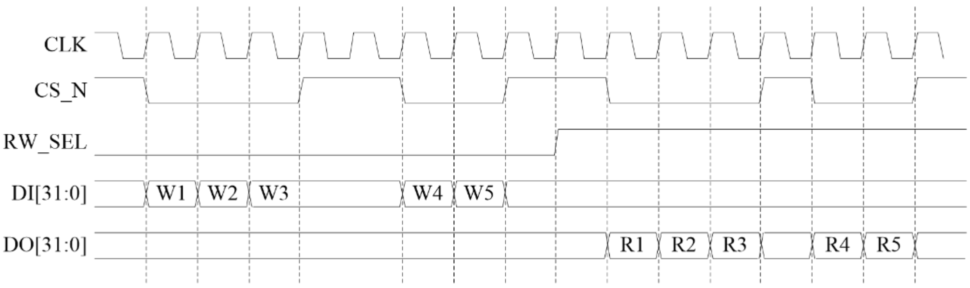


Figure 42 GTP_IPAL_E2 internal read/write interface timing from parallel interface

Note: RW_SEL signal jumping can only be done when CS_N is high.

4. Example templates

```
GTP_IPAL_E2 #(
        .DATA_WIDTH("X8").
        .IDCODE('b10101010101010101010101010101010101010101010101010101)
) <InstanceName> (
        .DO().
        .ECC_INDEX().
        .DI().
        .DERROR().
        .ECC_VALID().
        .RBCRC_ERR().
        .RBCRC_VALID().
        .SERROR().
        .CLK().
        .CS_N().
        .RST_N().
        .RW_SEL().
        .SEU_FRAME_ADDR().
        .SEU_COLUMN_ADDR().
        .SEU_REGION_ADDR().
        .SEU_FRAME_NADDR().
        .SEU_COLUMN_NADDR().
        .SEU_REGION_NADDR().
        .PRCFG_OVER().
        .PRCFG_ERR().
        .DRCFG_OVER().
        .DRCFG_ERR()
).
```

(II) UID interface

1. Port List

Table 41 GTP_UDID Port List

PORT NAME	I/O	BIT WIDTH	FUNCTION DESCRIPTION
DI	I	1	Serial Data Input
DO	O	1	Serial Data Output
SE	I	1	Data shift enable
LOAD	I	1	Data register parallel loading UID CODE
CLK	I	1	Clock

2. Parameter List

Table 42 GTP_UDID Parameter List

NAME	TYPE	VALID VALUES	DEFAULT VALUE	FUNCTION DESCRIPTION
UDID_WIDTH	Integer	96	64	UID length, for simulation only(1)
UDID_CODE	Binary	0~96'hffffffffffffffffffffffffffffffff	0	Chip ID, for emulation only

Note: (1) The interface model is common to Titan2 series devices, the UID lengths of different series are different, the default value is unified to 64, therefore, in order to keep the simulation consistent with the upper board, it should be changed to 96 when used for Titan2 devices.

3. Interface Timing

The UID length is 96 bits.

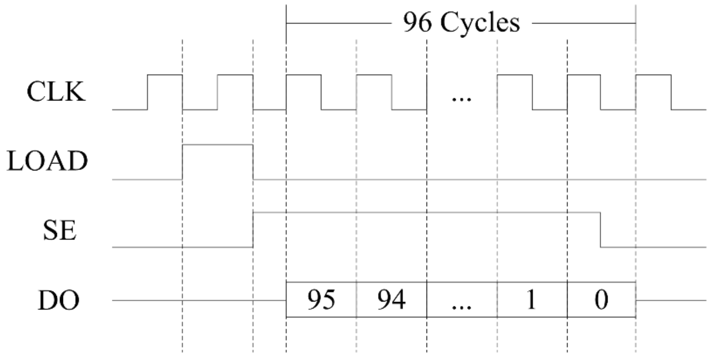


Figure 43 Read UID Timing

The user can extend the bit width of the UID.

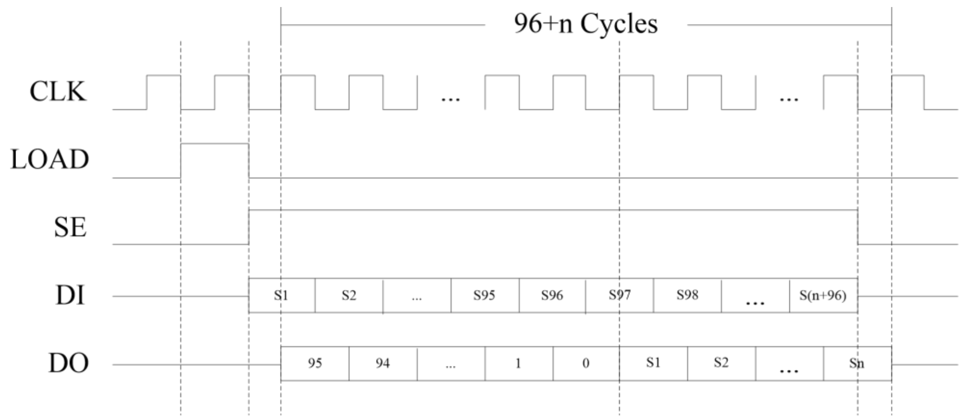


Figure 44 Extended UID Bit Width

4. Example templates

The Verilog routine template for GTP_UDID is shown below.

```
GTP_UDID #(
    .UDID_WIDTH(96)
    .UDID_CODE('b000000000000000000000000000000000000000000000000000000000000
    000000000000000000000000000000000000)
```

```
) <InstanceName> (
        .DO().
        .CLK().
        .DI().
        .LOAD().
        .SE()
).
```

(III) User JTAG interface

When the user JTAG interface is instantiated in the user logic, the user logic completely takes over the external JTAG interface. The interface timing and method of operation is identical to the external JTAG.

1. Port List

Table 43 GTP_JTAGIF port list

PORT NAME	I/O	BIT WIDTH	FUNCTION DESCRIPTION
TCK	I	1	User JTAG clock with a maximum frequency of 50M
TMS	I	1	User JTAG test mode selection. Control of TAP controller state jumps on rising edge of clock TCK
TDI	I	1	User JTAG test data input. Serial input to JTAG instruction and data registers on rising edge of clock TCK
TDO	O	1	User JTAG test data output. When the clock TCK falls, it is used as the serial output of JTAG instructions and data

2. Parameter List

Table 44 List of GTP_JTAGIF parameters

NAME	TYPE	VALID VALUES	DEFAULT VALUE	FUNCTION DESCRIPTION
USERCODE	Binary	32'h0~32'hFFFF_FFFF	32'hFFFF_FFFF	USERCODE

3. Example templates

The Verilog routine template for GTP_ JTAGIF is shown below.

```
GTP_JTAGIF #(
        .USERCODE('b11111111111111111111111111111111)
) <InstanceName> (
        .TDO().
        .TCK().
```

```
        .TDI().
        .TMS()
);
```

(IV) User Fuse interface

User logic can read user fuses by instantiating the user fuse interface (GTP_EFUSECODE), which outputs 32-bit user fuse values in asynchronous parallel fashion.

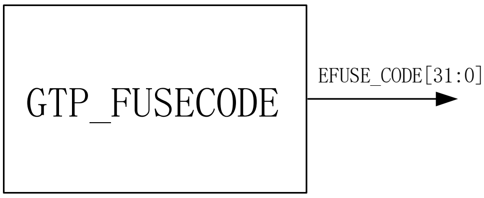


Figure 45 GTP_EFUSECODE Port Map

1. Port List

Table 45 GTP_JTAGIF port list

PORT NAME	I/O	BIT WIDTH	FUNCTION DESCRIPTION
EFUSE_CODE	O	32	Efusecode Output Data Bus

2. Parameter List

Table 46 List of GTP_JTAGIF parameters

NAME	TYPE	VALID VALUES	DEFAULT VALUE	FUNCTION DESCRIPTION
SIM_EFUSE_VALUE	Binary	32'h0~32'hFFFF_FFFF	32'h1234_5678	EFUSE_CODE Emulation value

3. Example templates

The Verilog routine template for GTP_EFUSECODE is shown below.

```
GTP_EFUSECODE #(
        .SIM_EFUSE_VALUE('b000100100011010001010101011001111000)
) <InstanceName> (
        .EFUSE_CODE()
);
```

VIII. Contact us

Контакты для технических и коммерческих вопросов

ООО «Инэк»

г. Санкт-Петербург, ул. Яблочкова, д. 20, литер А, оф. 504

contact@innek.ru

+7 (812) 200-40-37

Комментарии и предложения по переводу пишите на почту